Design • Build • Program • Learn • Compete

SERVO MAGAZINE

FOR THE ROBOT INNOVATOR

www.servomagazine.com

September 2012

♦ MATE ROV 2012

♦ ROS ON A CHIP

♦ Parallax ELEV-8 Quadcopter Build

TigerBot Earns Its Stripes As Research Platform

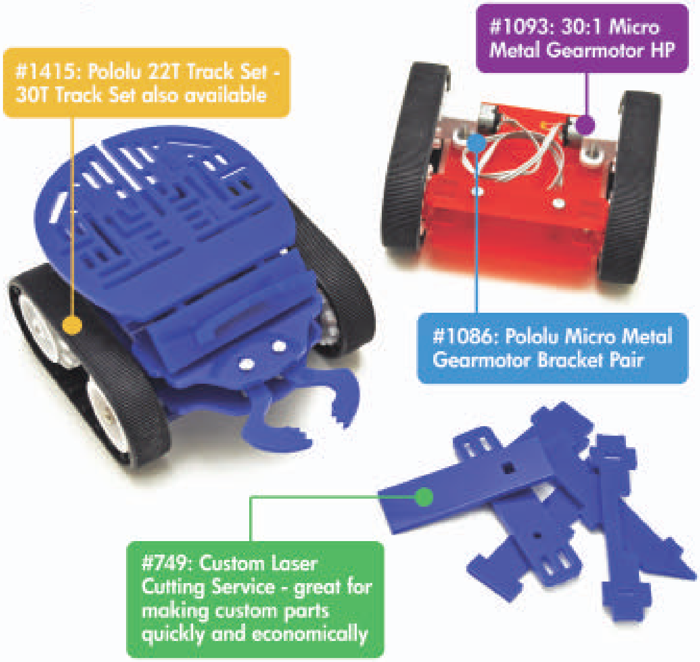
Award Winning Student Design

U.S. $5.50 CANADA $7.00

0 71486 02422 4 09

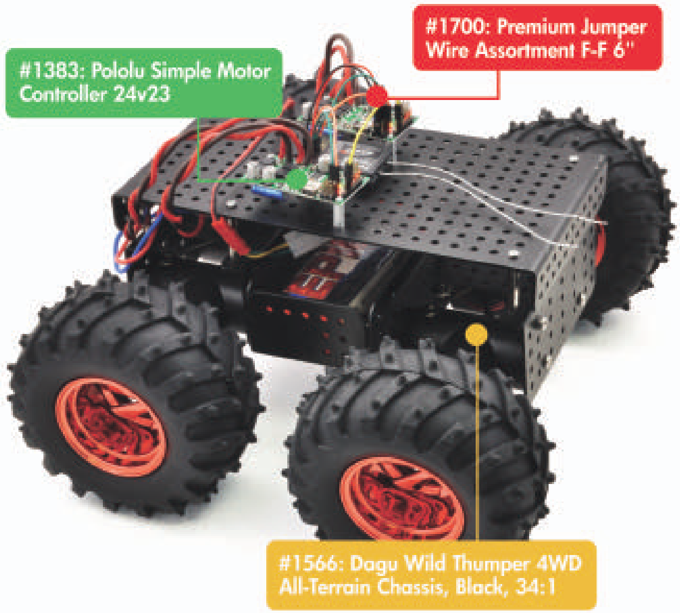
Custom Laser Cutting:

Design Your Own Chassis



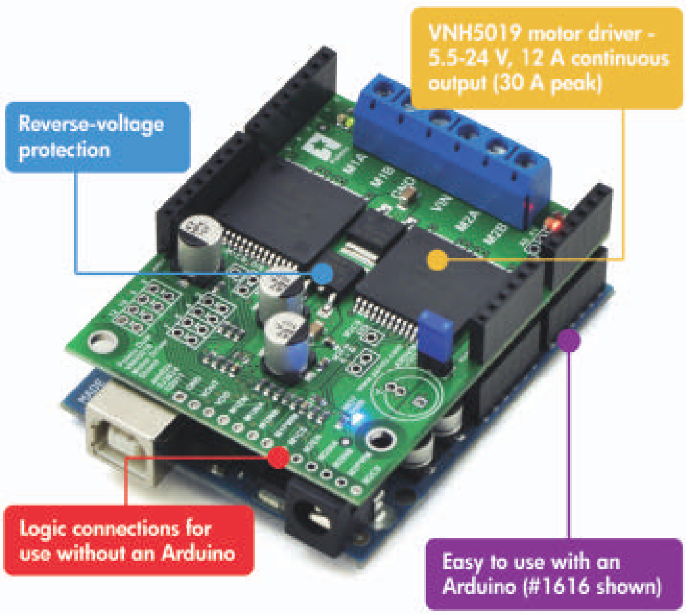
DIY Projects:

Wild Thumper-Based Robot



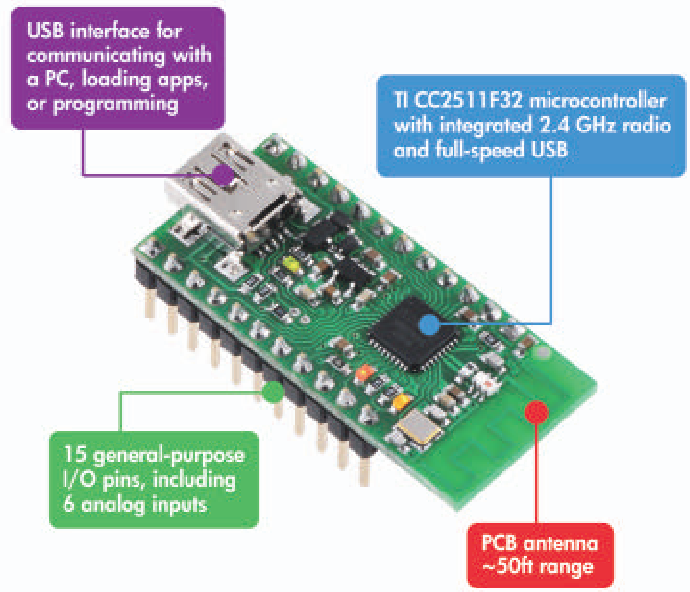
Motor Drivers:

Item #2502: Dual VNH5019 Motor Driver Shield for Arduino



Programmable Controllers:

Item #1336: Wixel Programmable USB Wireless Module



Finding the right parts for your robot can be difficult, but you also don't want to spend all your time reinventing the wheel (or motor controller). That's where we come in: Pololu has the unique products - from actuators to wireless modules - that can help you take your robot from idea to reality.

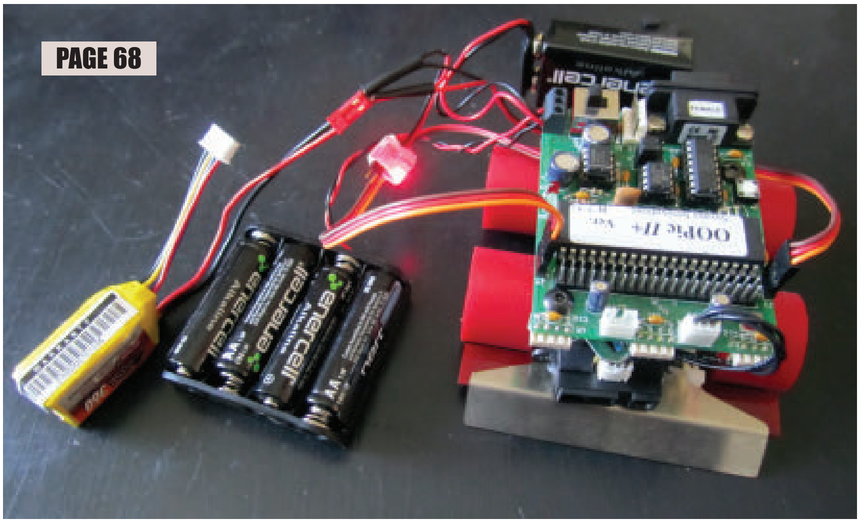
Pololu
Engage Your Brain

Find these products and more at **www.pololu.com**

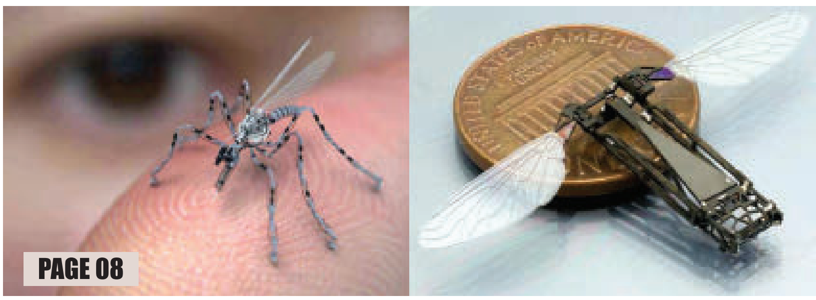
SERVO MAGAZINE

09.2012

VOL. 10 NO. 9



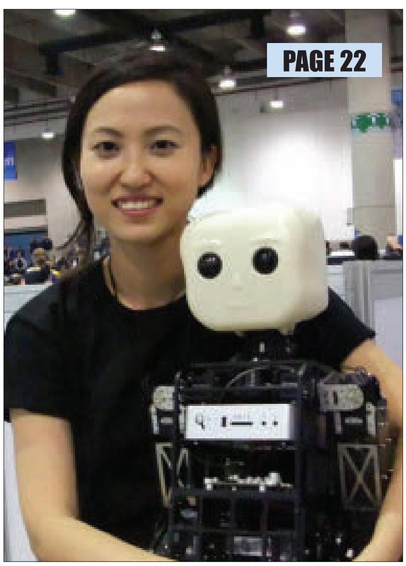
PAGE 68



PAGE 08

Columns

Departments



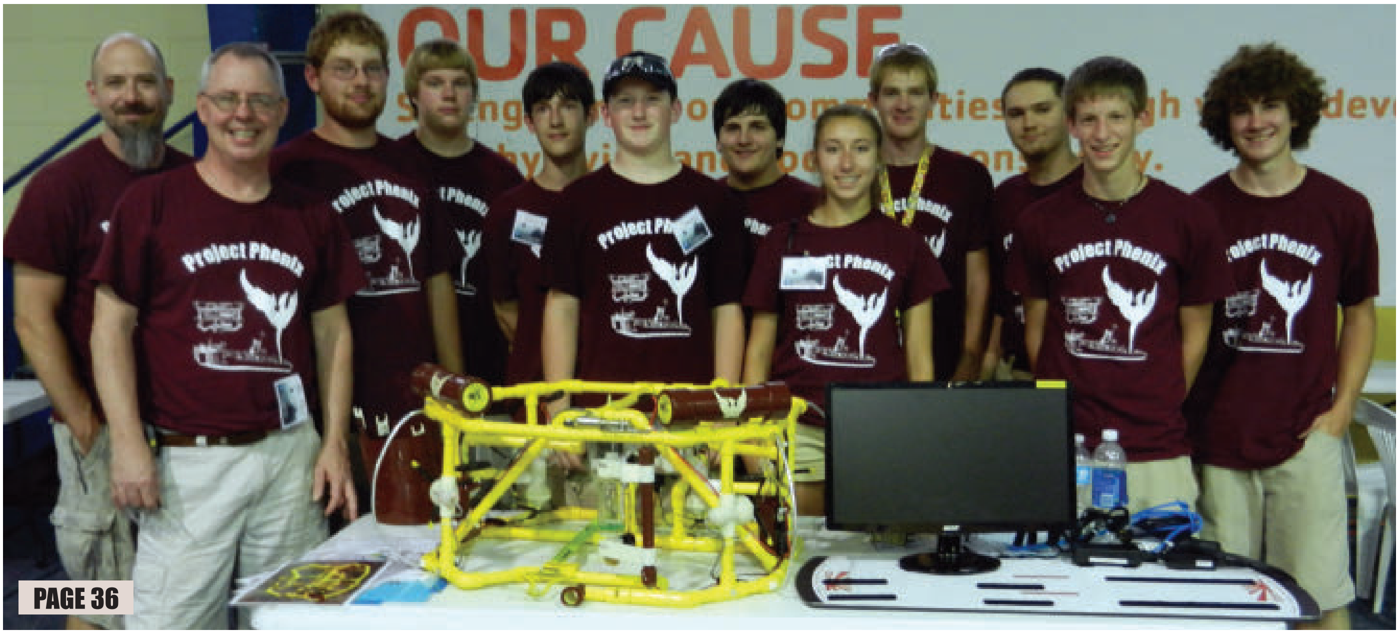
PAGE 22

PAGE 74

The Combat Zone...

SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...



Mind / Iron

by Bryan Bergeron, Editor

Danger, Will Robinson!

Robots are supposed to be great for the dirty, dull, and dangerous. Then on the flip side, for companionship and help around the house. However, it's easy to forget that robots can be perceived as — and sometimes are themselves — dangerous. It's something to consider when you work with robots around non-enthusiasts and when you're designing your next platform.

If you ever visit an automated factory with a robotic welding shop, you'll see that many of the machines are either fenced off or have a safety zone painted on the floor. Enter the zone or jump the fence and you'll risk serious injury. Most people expect a factory full of powerful, robotic arc welders and assemblers to be a dangerous place. The greatest practical danger to a traditional factory worker is often job security.

From a safety perspective, Google's driverless car — undoubtedly the future of driving — isn't there yet. The modified Toyota Prius is essentially in permanent driver's ed class, with a human emergency operator/observer in the car whenever the motor is running. Although legal in Nevada, like I said, the car isn't quite there yet. Until it can automatically, say, come to a stop when a five year old girl on a tricycle bounds out into the street, the car will be considered too dangerous for consumers. I suspect that the US DOD has several driverless transport vehicles on order, however.

If you've kept up with the work of the amazing folks at Willow Garage (**www.willowgarage.com**) — home of the Robot Operating System — you know that PR2 is probably the leading edge in personal robot appliances. It's an impressive robotics platform for serious academic R&D. I'd love for a chance to have access to one of these $400K machines. However, it's not something that I'd want roaming around my kitchen wielding a cleaver or helping me put on my tie in the morning. Those pincer hands just look too cold and powerful to me, and they must seem doubly so to someone not familiar with robotics.

Then, there's my latest experimental robotics platform of choice: the quadcopter. While small units can be flown indoors, they're at their best outdoors where larger craft sporting GPS and Google Maps and a variety of other sensors can be used to determine waypoints and photograph points of interest. Working with the larger, outdoor quadcopters is also a great excuse to get outside, away from the workbench. However, many city governments (and police) see

FOR THE ROBOT INNOVATOR SERVO MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Jenn Eckert, Tom Carroll, David Geer, Dennis Clark, R. Steven Rainwater, Kevin Berry, Pete Smith, Morgan Berry, Fred Eady, John Blankenship, Samuel Mishal, Bryce Woolley, Evan Woolley

CIRCULATION DEPARTMENT
subscribe@servomagazine.com

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION/GRAPHICS
Shannon Christensen
Sean Lemieux

BIO-FEEDBACK

Dear *SERVO*:

I just wanted to point out to you in the Sept '11 Mr. Roboto column that torque and force are not equivalent. Forces have units of Newtons and torque has units of Newton*meters. Motors should be rated in N*m (Force*Length) or N*mm of torque, but many of them are given in kg*cm or g*cm, etc. Note that this is the manufacturer's ignorance of the difference between mass and force, and they are rating their motors for the forces those masses produce as weight under Earth's standard surface gravity.

Also, sizing a motor to turn a wheel has nothing to do with sliding friction. The coefficient of rolling friction between a rubber wheel and concrete is more accurately on the order of 0.01. It may increase closer to 0.10 for older cars with multiple bearings and rough contact, etc. If it was closer to 1.0, it would require 2,700 lbs of lateral force to make a car budge, but I would estimate that it only takes about 50 lbs of lateral force, suggesting a coefficient of (rolling) friction around 0.02. Keep in mind that the formula $P(max)=(1/4)*T(max)*\omega(max)$ is an idealized approximation. In reality, each motor has a highly nonlinear torque-speed curve that looks more like a rounded hill than a flat linear (ideal) decline from max torque at zero rotational speed to zero torque at max rotational speed.

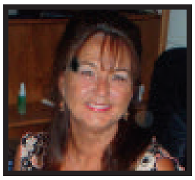
The final thing I'll mention is to remind people to use good engineering practice and build in a factor of safety, since all of our measurements for the components of these equations are estimates. I've found 1.5 works well for most robotics applications if the parameters are known well; 2.0 if estimates are fairly rough or if the application of the robot is still largely undetermined.

I do enjoy this magazine.

Jesse Maxwell
Mechanical Engineer
Metal Storm, Inc.

quadcopters and variants as large R/C aircraft that are already banned from flying down busy city streets. Even if your local community doesn't have laws about R/C model aircraft, as the operator you need to consider the personal liability involved if a two pound Lithium-Ion battery ends up smashing a car's windshield and then exploding.

Clearly, danger from robots — whether real or perceived — affects public acceptance of the technology. Eventually, we'll get to the point of having a robot in every home, perhaps warning us of impending danger from sipping a too hot cup of coffee. To reach that point, we'll have to think safety first in our robot designs. **SV**

Robytes

Discuss this article in the *SERVO Magazine* forums at http://forum.servomagazine.com.

by Jeff and Jenn Eckert

UUV Built for Consumer Market

The HydroView UUV from Aquabotix.

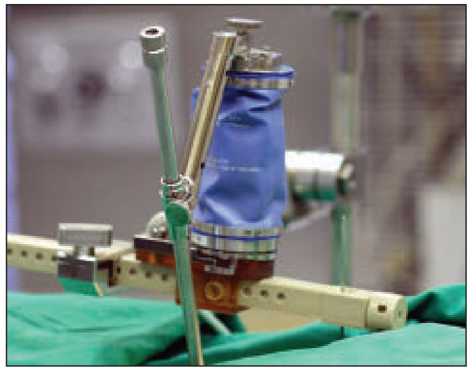
For years, the military, academia, and other institutions have been deploying unmanned underwater vehicles (UUVs) for various defense, inspection, and research applications, but submersible bots have been far too expensive for the consumer market. However, Aquabotix (**www.aquabotix.com**) intends to change all that with HydroView — "an affordable and easy-to-use underwater vehicle" that will allow you to play Jacques Cousteau in your local body of water. Under the control of any iOS device (smartphone, tablet, or laptop), you can satisfy your curiosity about what lurks beneath the waves, inspect vessels and structures, locate lost sunglasses and jewelry, or just annoy sea creatures for the fun of it. (Use in public swimming pools is discouraged, as it will probably get you thrown out.) HydroView shoots full 1080p HD still and video images, and the onboard LEDs provide visibility in murky or low-light conditions. The UUV is tethered to a floating topside box via a cable, allowing it to cruise around up to 5 kt forward and 1 kt reverse, operating at depths up to 150 ft (46 m). The standard cable is 75 ft, but custom lengths are available. The battery pack provides up to three hours of continuous exploration, and the 14.6 x 19 x 7 inch (37 x 48 x 18 cm) comes with a waterproof hard travel case. The bad news is that "affordable" is a relative term, since the HydroView Sport model will run you $3,995. If that's not in your budget, you might consider the company's AquaLens product which offers similar viewing and illumination features but mounts on the end of a pole instead of the sub. It's not nearly as much fun, but you can grab one for only $475.

The AirBurr, designed to navigate through cluttered indoor environments.

AirBurr v.8 Revealed

In 2010, we took a look at AirBurr — a micro air vehicle (MAV) under development at Switzerland's Ecole Polytechnique Federale de Lausanne Laboratory of Intelligent Systems (**lis.epfl.ch**). Its claim to fame is that, rather than employing some kind of sophisticated collision-avoidance system it just bounces off things and somehow keeps moving (much like my college buddies during a night on the town). Now, after two more years of evolvement, the Lab has announced that AirBurr has the added capability of righting itself and continuing on its journey even after crashing all the way to the ground (unlike the aforementioned buddies). It accomplishes this by virtue of a low center of gravity coupled with a spherical carbon fiber cage, plus four legs that extend from the body and push it into a vertical position. According to a paper published in *IEEE Transactions on Robotics*, the new Samurai model — which is the eighth generation of the device — can right itself within 25 seconds, 100 percent of the time. Well, assuming that it crashes onto a surface that slopes less than 10 degrees, that is, and doesn't include rocks or gravel, of course. AirBurr v.9 is already in the works, so maybe in a couple more years, those little constraints will have been eliminated.

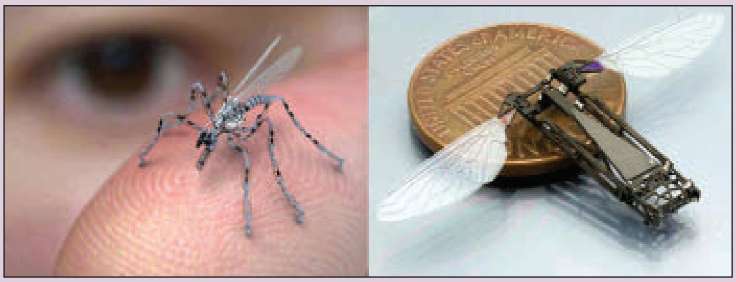
Yes, It *is* Brain Surgery



Mazor's SpineAssist bot, soon to be approved for brain surgery as well.

The SpineAssist surgical bots from Mazor Robotics (**www.mazorrobotics.com**), coupled with the company's Renaissance Surgical Guidance System have been used in more than 15,000 spinal implant procedures worldwide. The system has now been modified to perform brain surgery, as demonstrated in three biopsy operations at the HSK Hospital in Wiesbaden, Germany, thus paving the way for adoption in other countries. According to Mazor, both the US Food and Drug Administration and EU CE Mark regulators are reviewing the system for brain procedures and should be rendering their decisions later this year, allowing an official product launch early in 2013. Clinical trials were previously carried out on cadavers, but the latest are the first performed on live patients. Mazor CEO Ori Hadomi commented, "We are very proud about the first successful robotically guided procedures on the brain carried out in the world. This is the first and major step for the company and a technological breakthrough."

According to Mazor, the technology is applicable in the brain for biopsies, shunt placements, and neurostimulation electrode placement such as for deep brain stimulation (DBS). In the USA alone, about 25,000 brain biopsies are carried out annually, and the potential market for "inserting and navigating implants for deep brain stimulation therapy" is estimated at hundreds of millions of dollars.



Robo mosquito and butterfly. Are these real?

Robo Insects: Fact or Fiction?

In case you have missed it, there has been some scuttlebutt going around to the effect that the US military has perfected a spy drone that closely resembles a common mosquito. It is said to be so true to the original that you may not notice the difference. The remotely controlled bugbot is equipped with both a camera and a microphone and — as the story goes — can land on you, take DNA samples, implant an RFID device into your skin, or even inject you with toxins. The story is generally considered to be a hoax since the photo doesn't seem to depict all of those features, and there has been no confirmation from the military. There also has been no denial, but that is standard operating procedure. However, it is confirmed that a robotic butterfly with strikingly similar wings actually has been developed at Harvard University's Microrobotics Lab (**micro.seas.harvard.edu**), along with several other wing-flapping critters. So, we'll have to leave it up to you as to whether the mosquito is real or just a darn good prank. Either way, keep a fly swatter handy since Raid won't phase it.

Art Imitates Death

The Last Moment Robot sends a patient on her final path.

On occasion, we herein note the hazards of mixing artists with robotics, and a recent aesthetic flare-up has been spawned by Dan Chen, an "artist, graphic designer, interaction designer, web developer, and improvisational engineer." Dan's Last Moment Robot (LMR) is devised for bedside use in hospices, where it can serve as a replacement if family and friends can't be around to wave goodbye as you begin your eternal celestial dirt nap. The LMR caresses your arm and comforts you using a prerecorded script that runs as follows: "Hello [your name], I am the Last Moment Robot. I am here to help you and guide you through your last moment on earth. I am sorry that (pause) your family and friends can't be with you right now, but don't be afraid. I am here to comfort you (pause). You are not alone, you are with me (pause). Your family and friends love you very much. They will remember you after you are gone (pause). Time of death 11:56."

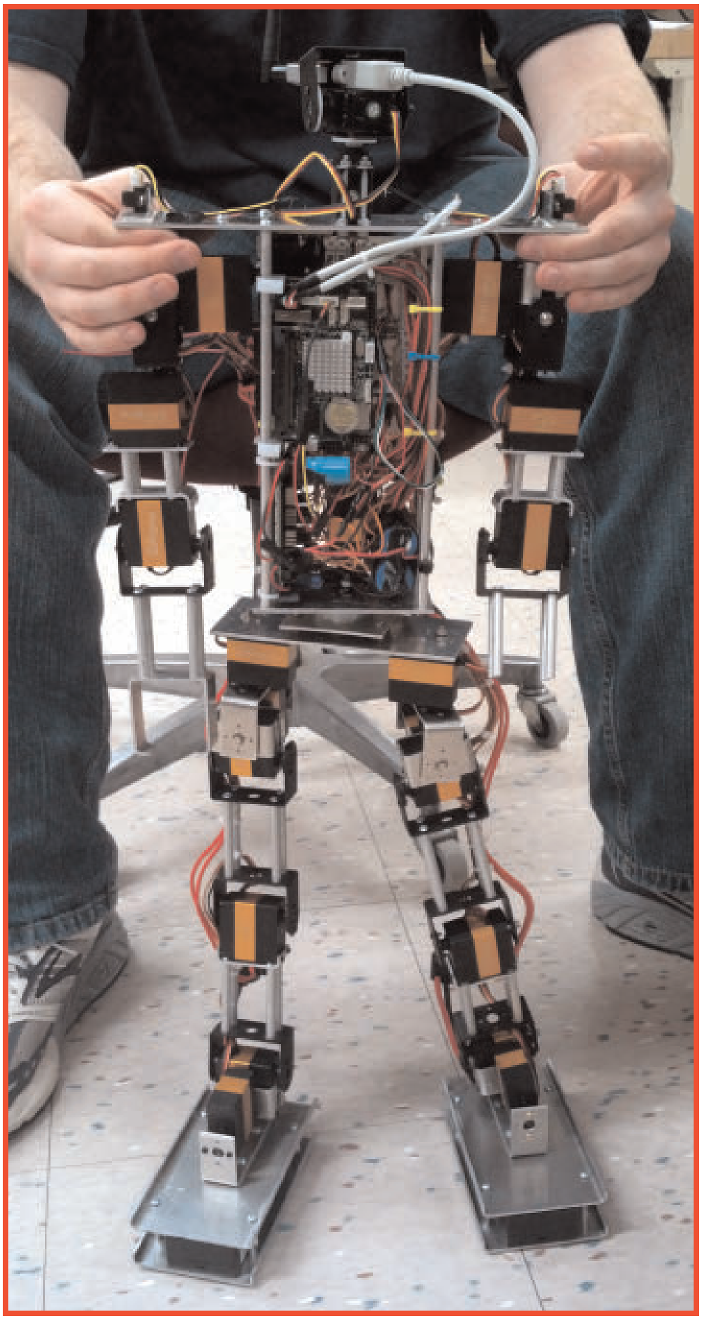
In Dan's defense, this exists only as an artistic demonstration and is not actually intended to be put to practical use. In fact, it is designed to question the appropriateness of such mechanical assistants as the Paro robot which is used in Japan as therapy for elderly and Alzheimer's patients (see *SERVO*, April '11). We get it. But it's still creepy. **SV**

by David Geer

Contact the author at geercom@windstream.net

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

RIT TigerBot: A Platform for Important Medical Research



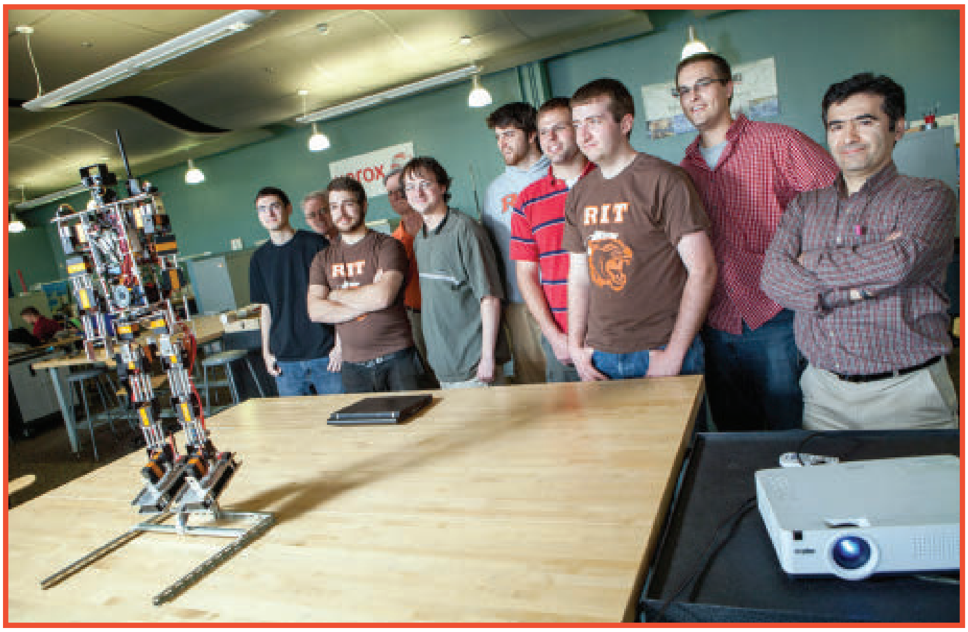
The autonomous humanoid TigerBot robot from RIT (Rochester Institute of Technology) recently won a design competition for the student chapter of the IEEE. The competition was comprised of students from colleges including those located in Syracuse, Buffalo, Rochester, and Boston. The 31 inch TigerBot was designed and produced during an 18 week course at RIT as a senior project.

TigerBot is an autonomous humanoid robot platform capable of interacting with users through speech recognition. TigerBot is an attempt at mimicking human movement and behavior. The purpose of this project was to create a robot capable of both human movement and interaction.

This is actually the second one of a series of projects planned for developing humanoid robots at RIT. The humanoid robot will serve as a test bench for future senior design projects regarding humanoid components such as an electromechanical body, intelligence, control, and sensors.

As mentioned, TigerBot is a 31" high automaton built to the scale of a human model. Along with mimicking human behaviors, the robot exceeded expectation in things like collision-avoidance, wireless and voice control, and interactivity. In fact, the plan is to have TigerBot — or some derivation of it — guide students through tours of the campus in years to come.

However, TigerBot will be primarily used for research. "Humanoids can be used to simulate lost limbs and be used for prosthetics," Kyle Backer (one of the team



The RIT TigerBot together with the whole robotics team.

members) explained.

The humanoid robot does have a ways to go before it begins giving tours to prospective students. In order for the robot to be functional, it will need a GPS unit among other aspects to be added by future senior design project students.

Combining Sciences to Create Robots

TigerBot was designed using skills from three engineering majors including mechanical engineering, electrical engineering, and computer engineering. Each aspect of the robot required input from students in each science major.

The mechanical engineering students designed the robot's joints and mechanical layout. The electrical engineers added to the joint design through the use of servo motors. The computer engineering students assisted with computer board layout so there would be space for the embedded electronics. The mechanical engineers also designed the robot to provide enough torque for several of the robot's high torque positions.

The electrical engineers designed the wiring and a power PCB to ensure the high power draw was appropriately used. They were also in charge of the placement of custom connectors for each component for better connections. The electrical engineering students designed the battery's size and storage so it would fit and still ensure a full hour of operation.

The computer engineers designed the programming of the robot, selecting the electronics that offered enough control for the robot while staying on budget. They interfaced the sensors with the embedded computer and also designed the balance algorithm. It was the computer engineers who interfaced all the communications between the electronics. They generated the programming platform and began the programming for the walking algorithms.

Ascending to Autonomy

In order to be autonomous, TigerBot had to be able to act independently of user control. The robot is completely unwired from battery power and command and control, and can walk around freely. Numerous environmental sensors and onboard computer intelligence instill it with further autonomy.

Sensors enable the robot to detect and interact with its environment and avoid objects, while the computer enables it to operate apart from manual command and control by an operator. The robot can recognize voices and apply sonar and IR sensors, which give it its environmental interaction and control intelligence. The voice recognition further enables the robot to respond and communicate with people in its direct vicinity.

Enabling Human Mimicry

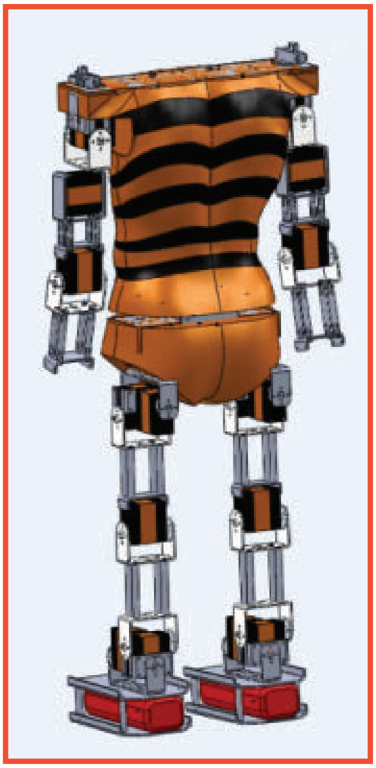
The robot has multiple systems that enable it to mimic human behaviors. It is scaled from human dimensions (as mentioned earlier), making it possible for the robot to look and function similar to a human being. The body layout is made up of aluminum rods which were machined by students.

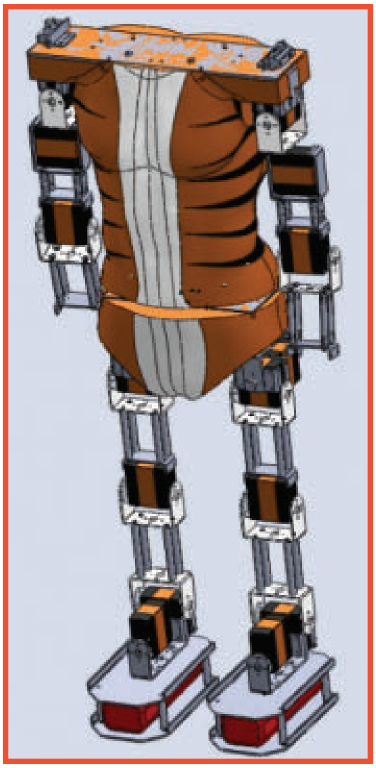
The robot has 23 DOF (degrees of freedom) of movement, including arms that bend at the wrist, shoulder, and elbow. The shoulders were designed to fully function as a ball joint, according to team members. The robot can pan and tilt at the neck, and the hips also simulate a ball joint. Finally, the legs bend at the knee, and the feet have vertical and horizontal rotation capabilities.

The degrees of freedom were enabled by RS-1270 revolute servo motors from RobotShop. These servos produce a rated torque of 480 oz-in. While the robot can make any motion a human can, it does not have a spine. However, this was not an obstacle.

Like a human being, the robot is autonomous so it doesn't have a cord to trip over, further enabling freedom to move as people do.

TigerBot has a USB camera head that feeds into the onboard computer for human-like machine vision. Future classes will explore environmental control using machine vision, according to the instructors. The robot has a speaker and voice control like a human ear, and can hear human





commands and follow them — another human function.

The embedded computer controls every aspect and capability of the robot. Like the human brain, it controls the robot's motions and responses to its environment. The embedded computer is a Roboard 100. Students wrote the programs that control the robot directly into the Roboard 100, skirting the need for external command and control. As noted, the robot was programmed in ROS.

Robot Control

TigerBot's wireless control consists of the onboard computer communicating wirelessly with other computers. The specific application of this is to use a wireless USB module attached to the computer to manually run the robot using programs that are external to the robot.

TigerBot's object avoidance is based on several sonar and IR sensors. The sonar sensors determine the distance an object is from the robot via sound; the IR sensors determine the distance an object is via infrared. By combining these sensors, the robot can determine how far away an object in its path is. Using its advanced onboard programming, TigerBot can avoid any object.

In addition to object avoidance, the robot interacts with people in its surroundings via voice control; it uses a speaker for its responses. An easy VRM was used to understand [verbal] user input. This is what gives the robot the intelligence to recognize voice commands. Once a voice command is understood, a set motion response can be triggered by that voice control command.

The vision system uses a camera with a USB communications modality which feeds visual input into the embedded computer, and then enables the robot to take images and video, and interact with the environment based on the collected images and what they mean.

The robot's capacity for balance (which keeps it from falling over) is based on an accelerometer and a gyroscope. These technologies interact with motion forces and with gravity. The data the technologies output is used to determine the motion of the robot with respect to its position, relative to gravity. With all this data, the robot has the input to balance itself, by itself.

Research Platform

TigerBot is a useful research platform for simulating disabled human bodies. A limb of the robot can easily be removed or turned off, so producing several human motions the same as a human being would without that limb can be explored. The researchers can scale down and test prosthetics on the robot to determine how the prosthesis responds during different types of motions.

In this way, TigerBot enables researchers to test many different versions of a single prosthesis long before human trials would start. They could study human motion after a limb has been lost or only crippled.

Looking Forward

Future students will be able to improve on the function of the robot. TigerBot has several areas for improvement including designing hands. The robot's control algorithms could become more advanced, such as for better walking and balancing. Since the robot was created within a budget, any area the roboticists could spend more on in the future could be improved. The robot could always use higher torque but with lower priced servos.

These college students created an autonomous robot with a lot of potential, especially in prosthetic research. Not bad for a senior semester. **SV**

Resources

RIT story about TigerBot
www.rit.edu/news/story.php?id=49204

Link to video of robot project
www.youtube.com/watch?v=RH8NFvrAunk

RS-1270 revolute servo motors
www.roboard.com/servo_1270.html

RoBoard RB-100
www.roboard.com/RB-100.htm

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

Our resident expert on all things robotic is merely an email away.
roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

by Dennis Clark

ASK MR. ROBOTO

I can't believe it is September already. Hopefully, you've all been busy preparing for or celebrating about some robot competition, of which there are many these days (yippee!).

Sometimes, we get tired of what we are doing and look for an upgrade to our robot. This month's question is just that — an upgrade. Anyone who has ever gotten one of the nifty biped robot kits has also gotten an IR remote to go with it. While an IR remote is fine for a TV that sits still at a fixed distance away in a room free from distraction, it isn't so great for a competition robot. I therefore delved into the realm of after-market upgrades using things never envisioned to be used like this. Curious? Read on then.

Q. I have a robot kit that I built that uses an IR remote control. The IR remote is okay, but not very reliable so I want to replace it with a radio control remote. I looked around and saw that the Japanese Robo One guys are often using a Sony Playstation 2 wireless remote for their bipeds. How can I get that to work for me and my robot?

— Dan

A. What a great idea, and why didn't I think of that? (Oh yeah, I don't play video games.) Still, what a great solution. The manufacturer has already worked out the RF details — it has to connect to a simple controller port, so this is an ideal solution for remote controls. When you have an idea that you are sure that someone has already come up with some solution for, the place to go is Google (or Bing, or Yahoo, or ...). I found a bunch of sites with some information on them and a couple of sites with lots of information on them, but none of them wrapped it up and tied the solution with a bow. It turns out this isn't rocket science, but it isn't super simple either.

Step 1 is to procure a PS2 wireless controller and get it to work with someone else's published software. Always start from a known position. That way, you can tell if you are doing something wrong or if something else is going on. As you know, I'm a Mac kind of guy so I went straight to a programmer platform that runs on the Mac and has *lots* of helpful people hacking on it. I'm talking, of course, about Arduino. Sure enough, someone has a PS2 controller library. Bill Porter looked over PS2 controller information and reworked some starter code written by another Arduino hacker on the Arduino forums. The result is this pretty decent library: **www.billporter.info/playstation-2-controller-arduino-library-v1-0**.

Strictly speaking, this library is just about talking to a wired PS2 controller. No matter, the wireless ones emulate the wired controller which is why we are so interested in it! Bill also has a useful troubleshooting guide for those having problems getting his library to work: **www.billporter.info/arduino-playstation-2-controller-library-troubleshooting-guide**.

I became acquainted with these pages when I didn't get my first PS2 wireless controller to work with the libraries. Before you go ***anywhere*** on this project, you really need to check out **http://store.curiousinventor.com/guides/PS2** which has lots of details you might find confusing, but lots of basic information you will need to wire up your controller.

I stand upon the shoulders of giants forging a path through the jungles! Many thanks to these and other folks I'll be mentioning further on as I give an account of my journey towards enlightenment. I do hope this article will get you moving along faster than I moved from start to finish to write it!

Okay, now that I have a library and a platform (Arduino) that has been vetted by others and is known to work, I'm off to get a wireless PS2 controller. I am very motivated by instant gratification, so I wanted to find a local store. I called all of the usual places: Kmart, Target, Walmart, Toys-R-Us, etc. Nothing. GameStop had a selection of previously used PS2 wireless controllers, so for $12 I picked up a nifty MadCats orange translucent controller that had a short extension cable to go between the console and the controller. That is the cable I'll be hacking into.

Creating the Connection Cable

The Curious Inventor site (as well as Bill Porter) had a quick diagram of the wiring of the controllers. They each mentioned wire colors which I quickly discovered were useless. Manufacturers of these devices clearly felt that consistency of wire color was for the weak. Everything I looked at had different cable wire colors, so we'll ignore the colors and go right to their position.

I wanted to use this short cable to interface to my Arduino and also use it later with other micros so that I can plug and unplug the RF transponder unit at will, rather than wiring it into the robot. The steps to do this are:

1) Cut off the male end of the cable (the one with pins, not holes).
2) Attach your desired connectors to the wires now exposed.

Simple right? Actually, it is.

Curious Inventor was all about wiring a PS2 controller to a robot — not to a wireless transceiver — so its wiring diagram is from the perspective of the male side which we just cut off. Lynxmotion sells wireless PS2 controllers AND the short interface cable — already with connectors on it. Their wiring diagram matches the cable end that we're going to want to wire up. To clear up confusion, **Figure 1** shows the wiring diagram from the perspective of the connector you'll plug your PS2 wireless transceiver into (similar to the Lynxmotion picture found at **www.lynxmotion.com/p-73-ps2-controller-cable.aspx**).

Click on the "connection diagram" link for the graphic showing wiring. I don't draw curves that well, so in my drawing what is actually a gentle curve is shown to be a sharp angle. So, think: "The 45 degree angle is actually a nice, radiused curve." Two signal lines aren't used by the wireless controller: motor power and N/C. The Ack signal isn't used by the Arduino library.

Remember, the wireless transceiver will have male pins; the cable we're wiring to our microcontroller will be the mate to that connector.

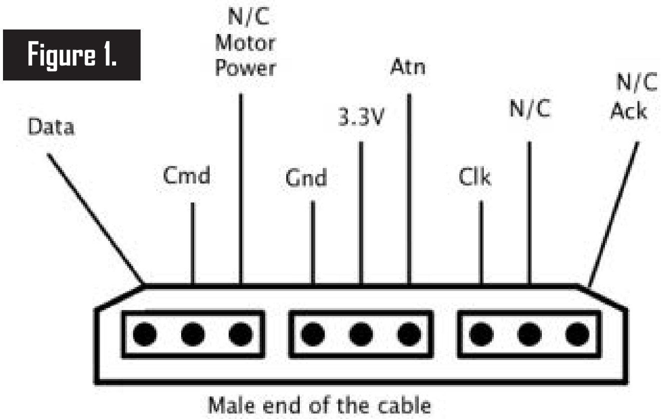
Connecting the Cable to an Arduino

Now that we have our interface cable built, we need to connect it to an Arduino board. To connect the PS2 controller to the Arduino, we need to pay attention to the names of the pins we have to interface. The example sketch that comes with the PS2x library is called "PS2x_example" — more on that later. The important part is that the default pins used for the example sketch are:

```
Attention (Atn) -> I/O 10
Command (Cmd) -> I/O 11
Data -> I/O 12
Clock (Clk) -> I/O 13
```

Stop right here! Many of the tutorials and discussions about interfacing the PS2 controller to a micro just connect the five volt power to the PS2 controller and the signals from their 5V powered micro directly to the PS2 controller, and press "go." My experiences have shown that this simply does not work. The controller wants 3.3V and sends back 3.3V based data. So that you can avoid wasting the time I wasted confirming this, I'll show you how to create an interface board that will handle the 5V to 3.3V level

Figure 1.



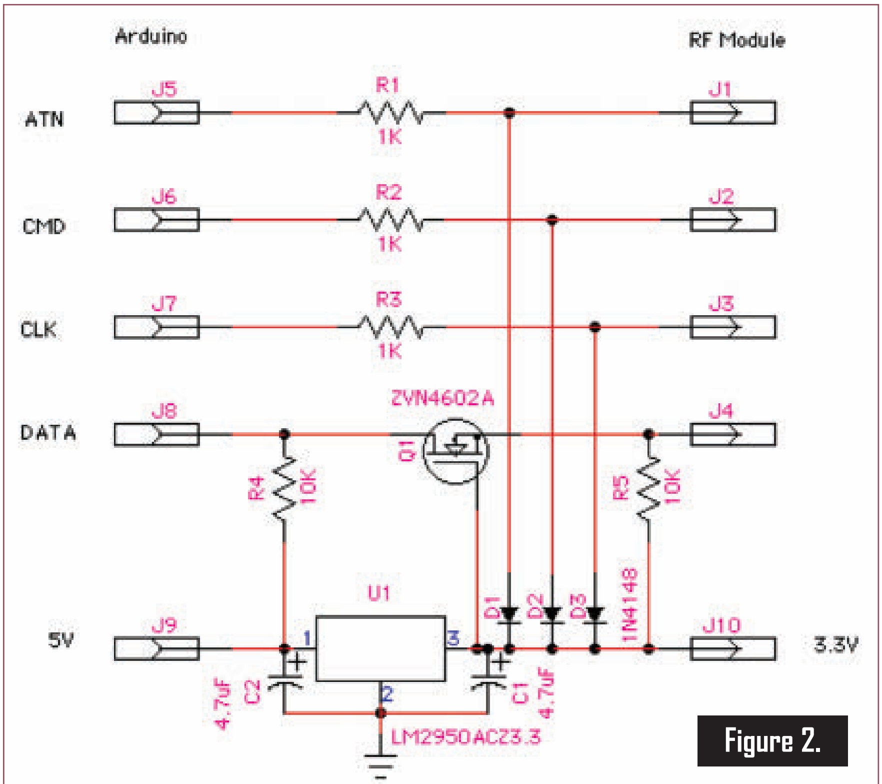


Figure 2.

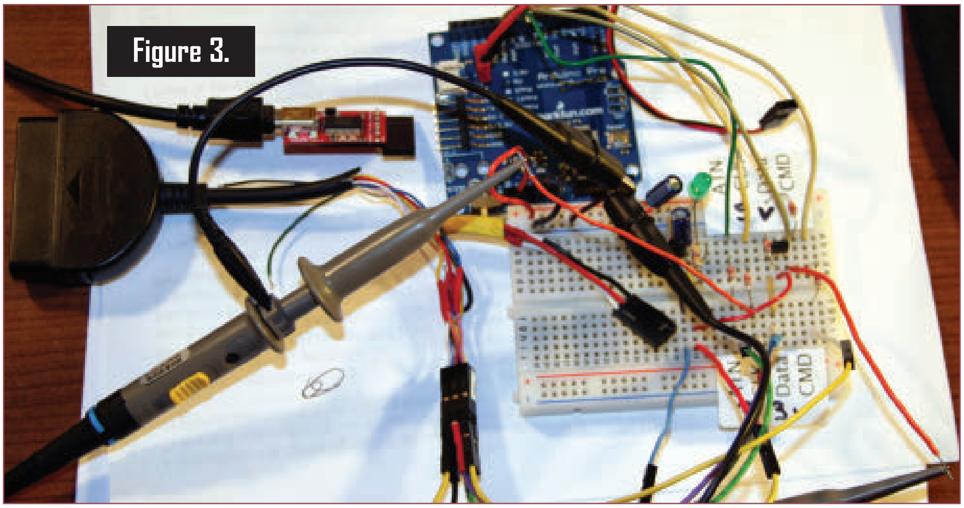


Figure 3.

translation so you can be ready from the start.

There are a variety of ways to connect 5V and 3.3V signals. Since the Arduino library connects to the controller at a clock rate of about 60 Kbps, we don't need to worry about creating high speed level translators; simple ones will do. **Figure 2** shows the circuit that I came up with that works just fine. Three of the signals go from the 5V Arduino to the 3.3V controller, and are quite simple. The Data line coming back from the controller is the one most likely to create a problem — a "can't read any data" kind of problem.

The first thing you need to provide is 3.3V to the PS2 controller. A simple LM2950ACZ 3.3 regulator is just the thing. With that and two 4.7 µF electrolytic capacitors, you have what you need. The ATN, CMD, and CLK lines use a 1K resistor and a small signal diode (1N4148) to shift the level. The diode clamps the voltage from the Arduino to the controller to 3.3V + 0.6V, which is safe for the micro in the PS2 controller. Many (if not most) micros have these protection diodes already in them, so the diode may not be needed. I don't know what the PS2 controller uses for its processor. Therefore, to be safe I simply included the diode in my circuit.

The 1K resistor limits the current that would go through the diode and keeps the current to about 3.3 mA per line to save on battery use. Larger resistor values would use less current, but they would run the risk of creating a voltage divider with any internal resistor of the PS2 controller's micro. This would create a bad logic level, so we'll use 1K ohms as a good compromise.

The last signal needs to be *up* shifted from 3.3V to 5V logic levels. I used a TO92 small current MOSFET to handle the task. In reality, this level translator will shift signals going either way so I could have used it for all of the signal lines. However, simplicity rules my design considerations, so I only used the MOSFET on the Data signal. Pay close attention to the diode polarity; the stripe is the cathode and that side goes to the 3.3V power line. The source pin of Q1 goes to the 3.3V controller side and the gate of Q1 goes to the 3.3V power rail.

Wire lead length isn't a big deal at these speeds, as you can see from **Figure 3** which is my setup where I made my level translator on a solderless breadboard. Label your wires so that you will minimize your mistakes.

Installing the PS2x Library for the Arduino

You can find the Arduino PS2x library here at **www.billporter.info/playstation-2-controller-arduino-library-v1-0**.

Unzip the archive and move the PS2X_lib folder to the Arduino libraries folder. On an OSX system, that would be in your user's Documents folder at **~/Documents/Arduino/libraries**. On a Windows system, it would be **MyDocuments\Arduino\libraries**. Re-start the Arduino and you will find the example program here at **File->Examples->PS2X_lib->examples->**.

Listing 1 shows the setup() section of the example sketch.

The pins chosen for the controller interface signals are arbitrary. The PS2X Arduino library is bit-banged and doesn't use a hardware SPI module. This is why its clock rate is only 60 kHz when the controller interface will run at 500 kHz. However, this means that you can put any function on *any* I/O line that is convenient, and it will work just as well. The setup() function will attempt to identify the controller you are using and its capabilities. The Arduino platform has some nice features to it; one of the nicest is the **Tools->Serial Monitor**. The Serial.println() function prints data out to that terminal so that you can see what is going on with your setup. This is a very handy feature.

Connect Your PS2 Dual Shock Controller to the Arduino

In a perfect world, you could now plug in your wireless transceiver to the cable you made, then load and compile PS2X_example, download to your board, open the serial terminal, and watch what happens. Unfortunately, we are not in a perfect world. You'll remember way back in this article, I mentioned getting a MadCats controller? I connected mine to my setup and it didn't work — not even a little bit. A buddy of mine, Joe and I got together to compare notes and he showed me his lash-up that worked. Hmm. We checked wires and connections (at that time, I didn't know about the 3.3V thing, and neither did he). After fiddling around, Joe mentioned that he needed a pull-up on the Data line to get his to work. So I did that, and still nothing.

Wishing to troubleshoot only my circuits, I borrowed his PS2 wired controller and it worked. I found the 3.3V

Listing 1: PS2X example setup().

```
void setup(){
 Serial.begin(57600);

 //CHANGES for v1.6 HERE!!! ***********PAY ATTENTION**********

   error = ps2x.config_gamepad(13,11,10,12, true, true);
   //setup pins and settings:  GamePad(clock, command, attention, data, Pressures?, Rumble?)
   //check for error

 if(error == 0){
   Serial.println("Found Controller, configured successful");
   Serial.println("Try out all the buttons, X will vibrate the controller, faster as you press
   harder;");
  Serial.println("holding L1 or R1 will print out the analog stick values.");
  Serial.println("Go to www.billporter.info for updates and to report bugs.");
 }

else if(error == 1)
   Serial.println("No controller found, check wiring, see readme.txt to enable debug. visit
   www.billporter.info for troubleshooting tips");

  else if(error == 2)
   Serial.println("Controller found but not accepting commands. see readme.txt to enable debug.
   Visit www.billporter.info for troubleshooting tips");

  else if(error == 3)
   Serial.println("Controller refusing to enter Pressures mode, may not support it. ");

   //Serial.print(ps2x.Analog(1), HEX);

   type = ps2x.readType();
     switch(type) {
       case 0:
        Serial.println("Unknown Controller type");
       break;
       case 1:
        Serial.println("DualShock Controller Found");
       break;
       case 2:
         Serial.println("GuitarHero Controller Found");
       break;
     }

}
```

comments on the Web and put together my level translator board. The PS2 wired controller worked great; the MadCats, nope. Okay, my pre-owned hardware was bad. I took it back to GameStop and exchanged it for a Logitech unit, and because the Logitech looked so "Darth Vader black cool" I got two of them. I took them home and connected them, they worked! Sort of. These initialized differently and the PS2X library could not set and lock the analog mode. Like the PS2 wired controller, all of the buttons could do both digital on/off and analog pressure returns. However, the horizontal joystick on the right did not work on either of them. Sigh.

I took both of them back and bought a new GameStop PS2 Predator S-Type controller. This one worked great and reacted exactly like the wired PS2 controller did! Exactly except for one thing — none of the buttons gave analog pressure data. No big, the joysticks worked fine. One thing that wasn't perfect was that the left direction button sometimes gave an indication of the L2 button being pressed, and pressing the L2 button sometimes didn't work. The L2 and Left Direction button are adjacent binary bits in the buttons byte; it appears that the PS2X library has

Figure 4.

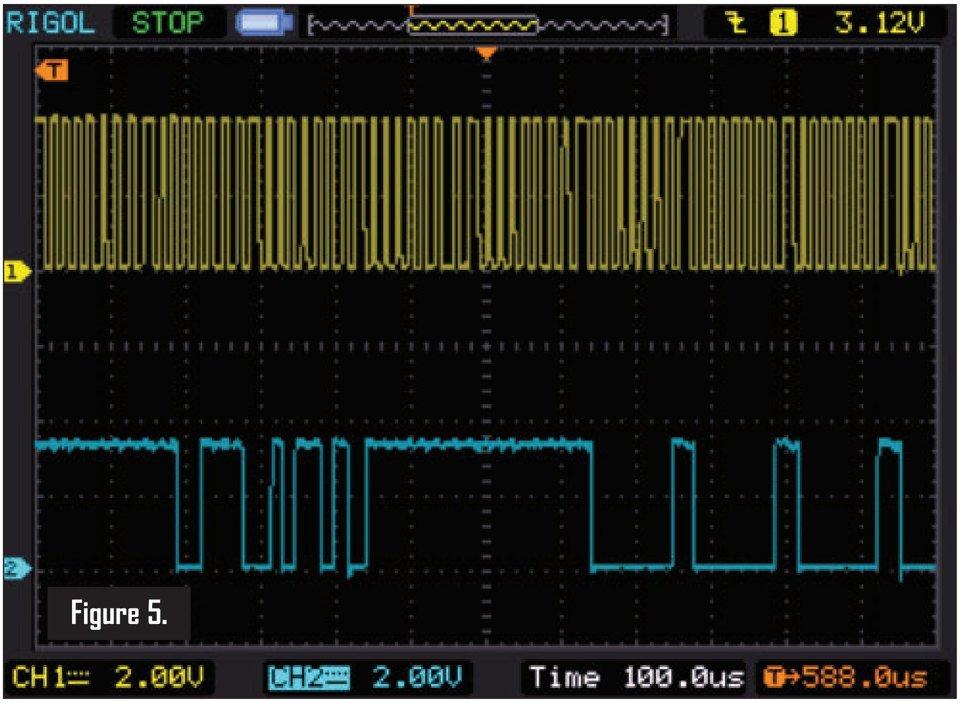


Figure 5.

a timing glitch with this. Not a big deal.

Figure 4 shows all the controllers that I tried. The silver GameStop controller isn't as sexy as the other ones and it is a little smaller, but it worked! I guess you could call this list "the good, bad, bad, and the ugly." Okay, maybe ugly is too harsh a word, but I've never liked my electronics to be silver.

The motto of this ordeal? Know your dealer to make sure you can return pre-owned hardware that doesn't work right (the GameStop folks were top-shelf cheerful about this) or buy only new. I bought a Lynxmotion wireless PS2 controller setup as well, but as of the time of this writing I hadn't gotten it yet. I'll let you know how it works and what the PS2X library discovers about it when I get it.

Now I have the hardware, cabling, software, and sketch working fine. **Figure 5** shows the Clock and Data trace of an entire dual shock wireless controller frame. The big "blips" to the right are three of the four joystick analog data bytes; the fourth one didn't fit on to the screen at this resolution.

BYTE	CMD	DATA								
1	1	Idle								
2	0x42	0x41								
3	Idle	0x5A	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
4	Idle	Data	Select	JoyR	JoyL	Start	UP	RIGHT	DOWN	LEFT
5	Idle	Data	L2	R2	L1	R1	Trangle	Circle	X	Square
6	Idle	Data	RJoyH							
7	Idle	Data	RJoyV							
8	Idle	Data	RJoyH							
9	Idle	Data	RJoyV							

Table 1. Analog Controller Mode.

Experimenting with one of these wireless setups got us at least 70 feet of range, then the local coffee shop wall stopped our range tests.

There's a breakdown in **Table 1** of the bytes you get back from a polling sequence in analog mode. The JoyR and JoyL are the buttons when pushing down on a joystick. RJoyH is right joystick horizontal; RJoyV is right joystick vertical. Same with the left side joysticks. Read the example program to see how one locks the controller into analog mode and reads the buttons if you can't wait for my next article.

It took me quite a while to get all of this together reliably, and I want to do justice with the programming sequences and use of the controller data to guide a robot. This will get you started, and you may even beat me to getting your robot running. I love this remote control concept and want to take it a bit further by using hardware SPI and cranking up the communications rate, but this is a really good start on getting your controller interfaced and working.

PS2 Controller Hacking Resources

I could not have gotten this far this quickly without the work done by many other hackers and innovators on the Web. This is my list of resources that were indispensible to this project:

- **http://sophiateam.undrgnd.free.fr/psx/index.html**
- **www.lynxmotion.com**
- **http://store.curiousinventor.com/guides/PS2**
- **www.billporter.info/playstation-2-controller-arduino-library-v1-0**
- **https://docs.google.com/document/pub?id=1gs5mdYlh5rhYV3daQOKFjRkavwM-pJjETpRCYLpKtHE** (You'll need a Google Docs login for this one.)
- **www.microchip.com** (helped with notes on level translation)
- And whatever pioneers were there before even these!

Well, that's it for this month. Next time, I'll be customizing this code a bit and hopefully moving it to a hardware SPI interface to make it faster and more robust. Until then, keep on hacking and building robots! As usual, if you have any questions for Mr. Roboto, feel free to email me at roboto@servomagazine.com and I'll be happy to try to answer them. SV

EVENTS Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**.

— **R. Steven Rainwater**

SEPTEMBER

11-14 **CIG Car Racing Competition**
Granada, Spain
Participants design autonomous controllers for robot race cars.
http://games.ws.dei.polimi.it/competitions/scr

15 **Robotour**
Czech Republic
Autonomous navigation in a park carrying a five liter barrel of beer.
www.robotika.cz

17-21 **World Robotic Sailing Championship**
Cardiff, Wales, UK
Robot sailboats must navigate an ocean course around buoys.
www.roboticsailing.org

21-23 **RoboCup Junior Australia**
Canberra, Australia
Events for several classes of soccer robots and rescue robots.
www.robocupjunior.org.au

OCTOBER

1-4 **UAV Outback Challenge**
Kingaroy, Australia
Search and Rescue Challenge, Airborne Delivery Challenge, and Autonomous.
www.uavoutbackchallenge.com.au

5-7 **MindSpark**
Pune, India
Events include Micromouse, robot dog fights, and robot search and destroy.
www.mind-spark.org

6 **The Franklin Cup**
Philadelphia, PA
Remote control vehicle combat.
www.nerc.us

18-21 **Latin American Robotics Competition**
Fortaleza, Brazil
Events include the Brazilian Robotics Competition, Robocup Latin American Open, and Brazilian Robotics Fair.
www.cbrobotica.org

19-21 **Critter Crunch**
Hyatt Regency Tech Center, Denver, CO
The original remote control vehicle combat event.
www.milehicon.org

NOVEMBER

23-25 **All Japan Micromouse Contest**
Toyosu, Koto-ku, Japan
Events for autonomous robots including classic Micromouse, half-size Micromouse, and robot race.
www.ntf.or.jp/mouse

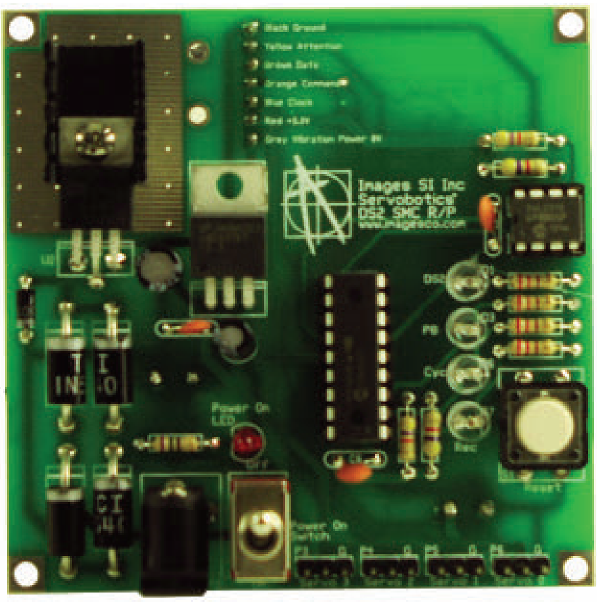
23-25 **Robotex**
Tallinn, Estonia
This is the largest autonomous robot competition in Estonia. This year's events include robot football, line following, mini Sumo, and LEGO Sumo.
www.robotex.ee

25 **Robocon**
Tokyo, Japan
Student teams from all over Japan come together at Robocon, where the robots they've designed compete in the Robo Bowl.
www.official-robocon.com

NEW PRODUCTS

PS2 Servomotor Controller Interface

The PS2 PlayStation record-playback servomotor controller interface from Images Scientific Instruments is a small DC powered interface board that will record and play back up to five minutes of servo motion for four standard 5 VDC hobby (R/C type) servomotors (Hitec/Futaba; not included).



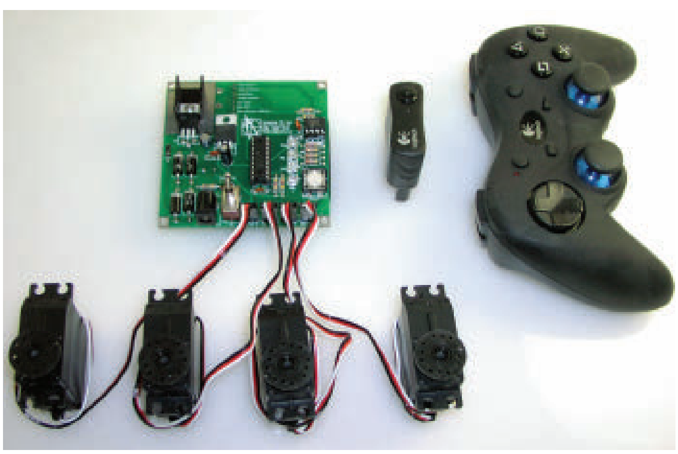
The four servomotors are controlled using the two joysticks on the PS2 controller. Each joystick controls two servomotors. The X and Y direction of each joystick controls a servomotor. Servomotor speed is proportional to the tilt of the joystick.

To record servomotor movements, a user presses a button on the PS2 to start recording. These sessions record the four servomotor channels simultaneously for up to five minutes. The PS2 is utilized to control the servomotors during a recording session. To stop recording, another button is pressed on the PS2 controller.

Once data is recorded, the user may play back the session with or without the Playstation controller attached. The controller allows for continuous loop playback of all servomotor movements recorded.

No programming or host computer is required. The PS2 interface is a stand-alone device that uses the PS2 to record and play back both movements and speed. A PS2 controller is not required for playback.

For further information, please contact:



Images, Co.	Website: **www.imagesco.com/servo/pssmc-m.html**

Aluminum Gearbox Arms

In addition to their current line of ABS gearbox arms, ServoCity has recently added two new aluminum gearbox arms: a 4" single arm and a 6" double arm. These all new aluminum gearbox arms are constructed of 1/4" thick 6061-T6 aluminum for superior strength and durability. The aluminum arms contain the 0.770" hub pattern which allows easy attachment to any of their servo power gearboxes, including the new Mega Servo using 6-32 screws. These aluminum gearbox arms are perfect for R/C sailboats, power boats, large scale robots, or any high

torque application that requires a solid gearbox arm.

Quarter Scale ServoBlocks

ServoCity also introduces their all new patented Quarter Scale ServoBlocks™ which increase a servo's load-bearing capabilities by helping to isolate the lateral load from the servo spline and case. The versatility of ServoBlocks allows users to create complex, extremely rigid structures with ease using standard Hitec servos. The 1/2" aluminum hub shaft provides multiple mounting options using 6-32 screws. The robust 6061 T-6 aluminum framework acts as a servo exoskeleton, greatly enhancing the mechanical loads the servo can withstand. ServoCity's new .770" hub pattern is repeated throughout the framework to allow endless attachment options.

The new quarter scale ServoBlocks are compatible with the following Hitec servos: HS-755MG, HS-755HB, HS-765HB, and HS-785HB. When used with a Hitec HS-785HB servo, the user can achieve 3.5 rotations without any modification. The kit ships unassembled and the servo is not included.

For further information, please contact:

ServoCity	Website: **www.servocity.com**

Continued on page 63

bots IN BRIEF

FUN IN THE SUN

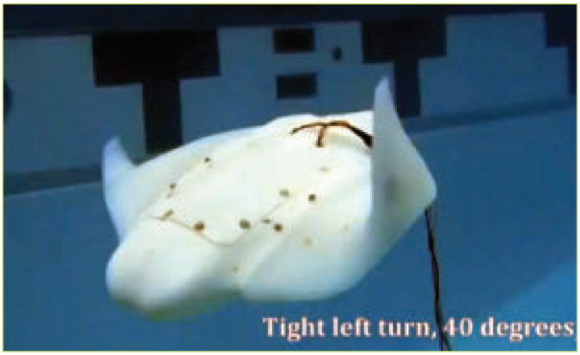
Solar panels are obeying the will of Moore's Law by getting ever cheaper and more efficient. What's not getting cheaper or more efficient is the human labor required to install them. This keeps the cost of going solar higher than we would like, but robots are busy coming to our rescue by setting up solar power plants much cheaper and much, much faster.

For example, using robots to set up a 14 megawatt solar power plant can potentially cut costs anywhere from $2,000,000 to $900,000, while being constructed eight times faster with only three human workers instead of 35.

The robot that performs this incredible feat of engineering efficiency costs just under a million bucks, but it's built from off-the-shelf parts, and in continuous use will supposedly pay for itself in either no time at all or less than a year (whichever comes last). Like all robots, using one of these things means you can get work done in rain or sleet or snow or darkness with no complaints. However, if you find yourself installing solar panels where all of those things are occurring, you might want to go someplace else. You know, sunny.

The robot itself has a mobile base that runs on tank treads, and a robot arm grips huge 145 watt panels one at a time and autonomously positions them in just the right spot on a pre-installed metal frame. Humans follow along behind, adding fasteners and making electrical connections, but secret plans are underway to roboticize these jobs too. Germans — being big fans of solar power in their quest to go 80% renewable by 2050 — are quite interested in putting robots like these to work as are the Japanese, who want to construct solar farms near Fukushima within the next six months.

A RAY OF HOPE



Bioroboticists at the University of Virginia (UVA) have built themselves a robotic cow-nosed ray.

Why? Because they can.

Also because rays are great at what they do, and if we can copy all their tricks to make better underwater robots, we absolutely should.

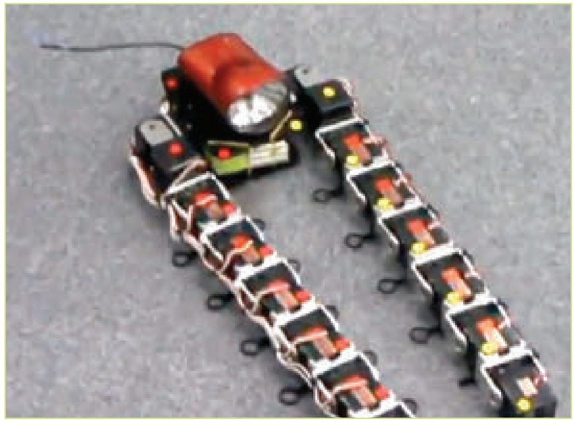
It's no coincidence that all the coolest UAVs look like rays. The form factor that was invented by batoidea eons ago is advantageous for a number of reasons common across fluids including both air and water, including high efficiency, good maneuverability, speediness, and lots of payload space. In other words — according to the UVA researchers — rays are "wonderful examples of optimal engineering by nature."

UVA's bioengineers aren't the first roboticists to notice how awesome rays are at being all ray-like. Festo (which knows a thing or two about robots inspired by nature) made both aerial and aquatic versions of rays that are quite acrobatic. What UVA is doing differently, however, is focusing on all the subtle ways that aquatic rays can control themselves, with the idea of developing an underwater robot that can do the same thing.

Making turns like rays do is an ability completely unique to the ray design, and it's a great illustration of why bioroboticists are so interested in getting all the details right. The body of the roboray is made of plastic, while the wings are made of silicon stuffed with rods and cables that expand and contract to cause the wing to change shape in ways that are modeled directly on observations of live rays.

The end goal here is an autonomous underwater vehicle that will be able to silently blend in with other sea creatures, carrying environmental monitoring payloads or possibly spy gear for the military.

METAL WORKING



This robot called "Metallic Vaio 2012" — straight from Japan — has a style of locomotion that we've never seen before. Instead of using arms or legs, it's got a sort of combination of both: two long tentacles made out of chains of servos that it uses to crawl around and rapidly somersault from place to place.

This robot was built (or should we say invented) by Eiichiro Morinaga, the guy who founded the ROBO-ONE bipedal humanoid competition. Besides its name, we know that it apparently has 18 degrees of freedom, and that it was designed to compete in the 6th KONDO LAND Multi-Legged Robot Obstacle Race where it took second place.

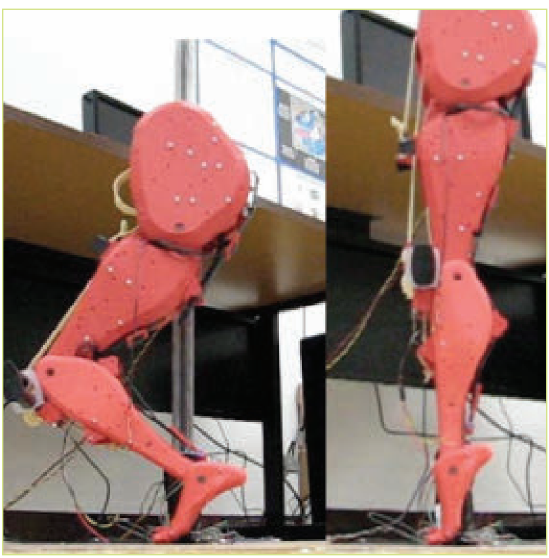
While Metallic Vaio 2012 may not be the most efficient of robots, Morinaga has certainly come up with something unique and quite capable by the looks of it. Adding a simple manipulator to the ends of those tentacles, for example, would create a robot that could use all those degrees of freedom to grasp stuff as well as to move, although doing both at once would be a little tricky. One solution might be to just add more tentacles (always a good idea), and sooner or later you'll end up with that octopus robot you've always wanted.

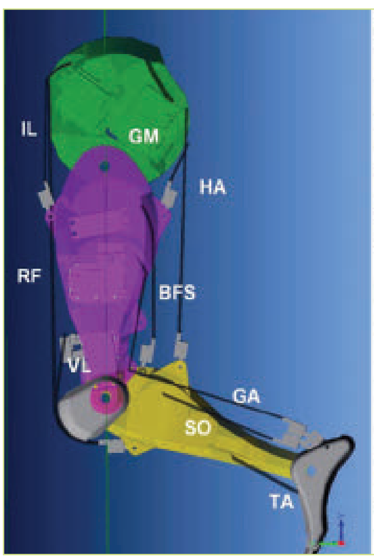
LEGS TO STAND ON

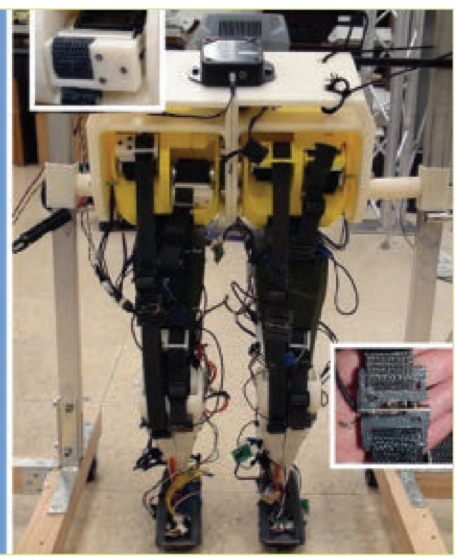
Dr. M. Anthony Lewis, Director of the Robotics and Neural Systems Lab at the University of Arizona and Theresa J. Klein (PhD student) have been working on a biarticulate muscle leg model. In a paper published way back in 2008, they described how motors pulled on stiff tendon-like Kevlar straps to reproduce the action of key muscle groups.

Their new biped robot features an improved leg design that models even more muscles, and it's already walking (though it relies on a babywalker-like support for balance). It stands 55 cm (22") tall with the legs fully extended and weighs approximately 4.5 kg (10 lbs).

A relatively simple motor controller based on a central pattern generator (CPG) produces a rhythmic output, causing the muscles to essentially flex back and forth. The amazing thing is that a naturalistic walking gait emerges dynamically from the interaction between its musculoskeletal architecture, its reflex system, and the CPG. Their research suggests the CPG stabilizes the walking gait against disturbances.





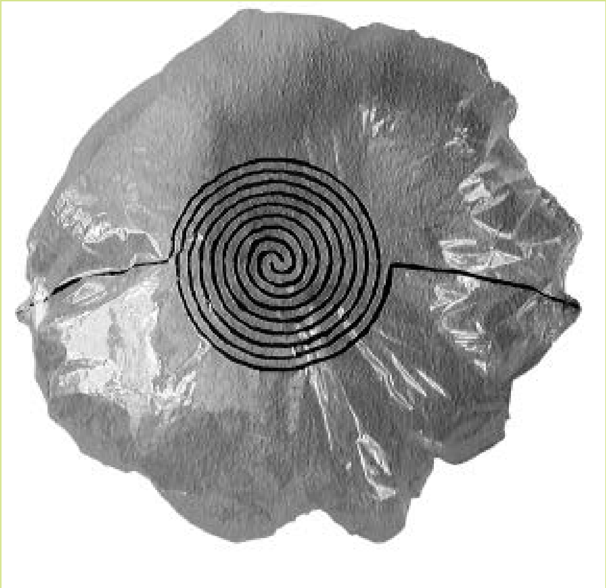


Artistic rendering of the project: soft modules floating in air, forming an artificial multi-cellular organism.

ROBOT FLOAT

This is a project that's being developed at EPFL (École Polytechnique Fédérale de Lausanne). The Laboratory of Intelligent Systems (LIS) is working on a robot made up of soft, floating modules that connect to each other through electroadhesion.

Electroadhesion is engineering magic that works by using very high voltages to generate a charge differential between two surfaces, causing them to stick together. The nice thing about electroadhesion (besides the fact that it works even on non-conductive surfaces) is that it's flexible, making it an ideal dynamic connector for soft, modular robots. Where EPFL is really going nuts, though, is with these soft robotic modules that float.



(Left) Two modules connected; (right) Module mockup featuring electroadhesion.

Modular robots are capable of adapting their morphology to tasks and environments which makes them more versatile, flexible, and robust compared to fixed-bodied bots. Most current systems lack mechanical flexibility when increasing the number of modules due to hard building blocks (modules) and highly rigid connection mechanisms. Although this design guarantees controllability and stability, it minimizes flexibility. In order to improve adaptation to environmental changes, softness on the module level might be beneficial.

The goal of this project is to look at how mechanical module softness can increase the efficiency and the capabilities of a modular robot. However, coping with softness requires rethinking the way modules are built. This study will be carried out with soft modules that feature a reversible connection mechanism, active deformation, and sensing. It will include design and development of novel soft technologies, smart sensing, and actuation. Go to **http://lis.epfl.ch/SoftRobotics** to find out more.

BED BOT

Too lazy to make your bed? Then this should be your next purchase. Spanish furniture makers OHEA has devised the Smart Bed that can make itself in 50 seconds. It will do this after three seconds of being empty when set on automatic mode. A mechanical arm rolls the covers up to the top of the bed while the pillows are straightened and set back down. Price was not listed but we expect it's one of those cases that if you have to ask how much it costs, then ...

There are also many instances where because of advanced age, some type of physical disability, or because of an accident, the individual may simply be unable to make a bed, so this might not be so frivolous, after all.

BOT IN STORES NEAR YOU

Underneath that color-coordinated hoodie is AndyVision — Carnegie Mellon's inventory assistance robot. It's programmed to take over the drudgery of daily retail inventory, helping stores figure out what customers want.

You may recognize the Kinect sensor underneath AndyVision's hoody, and he's also got a fairly simple mobile base with sonar for obstacle avoidance. Using Kinect, AndyVision scans store shelves to count items for inventory (using contextual object recognition), and will wirelessly alert store staff to low stock, no stock, or items that have been misplaced. Meanwhile, customers get access to real time data on what items are where and how many are left. The technology involved isn't anything new and crazy, but it's a great example of a relatively simple robot being used to do valuable autonomous work in a commercial environment.

AndyVision is a project from the Intel Science and Technology Center (ISTC) at CMU, and is part of a "Retail 2020" project to "transform the retail landscape." He's fairly retail-futuristic as is, but ISTC has other plans for the future where "in-store robots might handle tasks such as folding clothing items, stocking shelves, and helping customers to locate items and load their purchases into their cars."

LOST AND (STILL NOT) FOUND

The search continues to find the location of Amelia Earhart's plane after finding an old photograph of Nikumaroro Island in the Republic of Kiribati with something "suspicious looking" in the water. PIH's autonomous Bluefin Robotics 21 will be searching in this area by means of multi-beam sonar. A second dive will involve black and white photography with the team collecting data, replacing batteries, and reprogramming when needed.

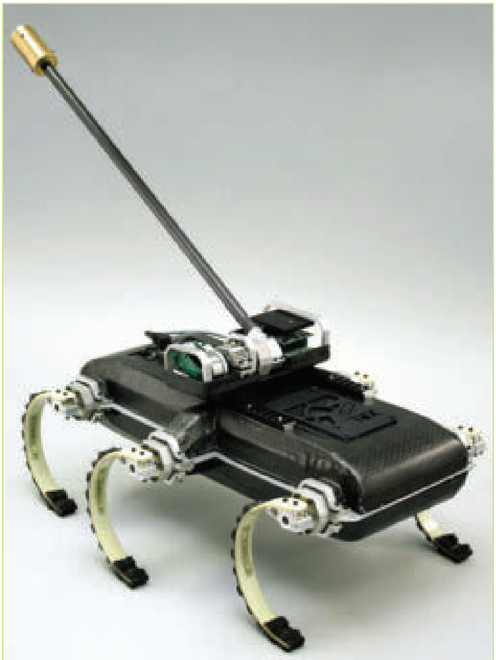
A TRV 005 robot with manipulating arms made by Submersible Systems will try then for close-up views with a high-def video camera to be controlled by a human on the surface ship. The project is being funded by TIGHAR (International Group for Historic Aircraft Recovery), after raising almost $2.2 million from various sources, including the US State Department and private companies. They believe that Earhart and navigator Fred Noonan may have landed on the reef of a coral atoll.

The expedition began July 2nd from Honolulu when the Hawaiian research vessel Ka'Imikai-o-Kanaloa departed. The date marks the 75th anniversary of Earhart's disappearance. Go to **http://tighar.org** for updates.

GETTING THE FINGER

Researchers at the University of Southern California's Viterbi School of Engineering have succeeded in making an artificial fingertip that outperforms humans in identifying a range of textures. That fingertip — the BioTac® from SynTouch LLC — is a molded elastomeric sleeve with a fingerprint-like pattern on the outside and sensors on the inside, filled with a conductive fluid. What the USC researchers have done is to develop algorithms for interpreting the data produced by the fingertip, and for optimizing the movement of the robotic arm or hand on which it is mounted to most efficiently produce useful data. Their findings have been published in *Frontiers in Neurorobotics*. SynTouch LLC (founded in 2008) "is a start-up technology business that develops and manufactures tactile sensors for mechatronic systems." BioTac sensors are available as an evaluation kit, and also as kits for the BarrettHand and the Shadow Hand.

BioTac's patented design consists of a rigid core surrounded by an elastic skin filled with a fluid to give a compliance remarkably similar to a human fingertip. BioTac is the first sensor capable of detecting a full range of sensory information that human fingers can detect such as forces, microvibrations, and thermal gradients.



BOTTY WHIP

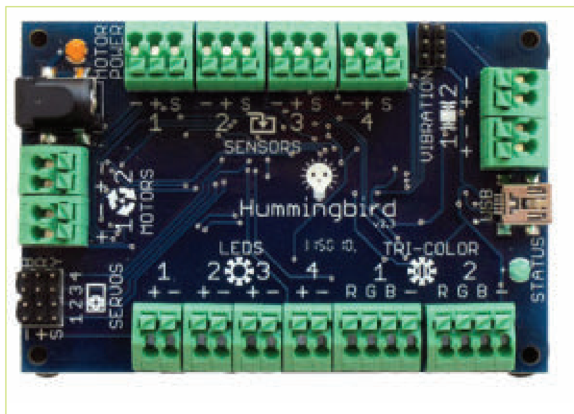
As it turns out, robots with tails can fly through the air while maintaining their orientation, and now other robotic platforms are testing out this technique, thanks to a collaboration between UC Berkeley and the University of Pennsylvania.

The bot you see here is X-RHex Lite, or XRL for short. It should look a little bit familiar (just like EduBot and SandBot) since it's based on RHex — UPenn's original legged hexapod. Except this guy is a lighter and more modular design. The new bit is, of course, the actuated tail which gives XRL the ability to right itself in midair when dropped at weird angles, as well as a way of maintaining its orientation if it runs off a ledge.

XRL tips the scales at 8.1 kilograms, making XRL about 60 times more massive than the original Tailbot. Note the comparison pic below.

There's a huge amount of potential for mobile robots with tails, and XRL marks a transition point between proof-of-concept and potentially operational platform. XRL is also particularly suited to midair shenanigans due to its six springy legs which act like excellent shock absorbers (as long as the robot lands on them the right way).

Adding a big long tail with a weight on the end plus a hefty actuator increases the complexity of robots like these, but there are ways to turn this into a good thing. For example, you could use the tail as an antenna or a mast for a camera.



HUM(MINGBIRD) DINGER

Robots are intimidating and starting from scratch with them is hard, no matter what age you are. You usually have to learn both hardware and software at the same time to get a robot to do anything cool, and for people without a background in either of these things surmounting that initial learning curve can be scary.

BirdBrain Technologies — a spinoff from Carnegie Mellon's Robotics Institute — has just released a new DIY kit called Hummingbird that promises to make building a robot as easy (and affordable) as possible.

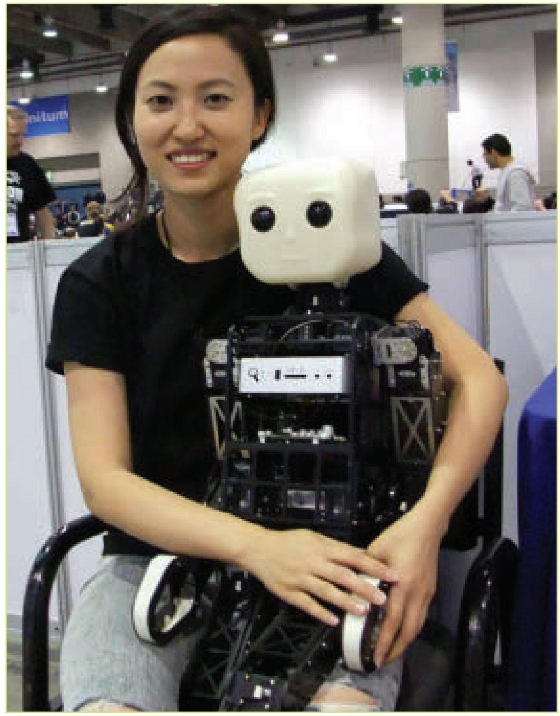
As you'd expect, the Hummingbird kit involves both a hardware component and a software component. Everything's included, with a clearly marked board and color coded wiring. It's also nifty that the wires just snap in and out — no soldering required — although soldering is not that hard and building simple robots is a great excuse to learn how.

On the software side, the kit comes with a Java-based drag-and-drop visual programming interface that doesn't require any previous experience at all, and anyone with a passing obsession with their iPhone should be able to get it working in no time.

Although this is called a kit, there's not instructions that tell you what to build. You use your imagination and some creativity to build a robot of your very own. You might need some additional structural components (like cardboard), but beyond that all it takes is a good idea to make whatever you want (which is what's so great about robots in general).

The Hummingbird kit is intended for kids of ages 10 and up, although it's not a bad way for people of any age to get familiar with making hardware and software work together. At $199 each, it might be a little more realistic to see the kit become part of an educational curriculum as opposed to something that kids will be able to buy for themselves. If you've got a budding roboticist in your family, though, we'd say this is probably a good investment.

COPEDO-ING WITH TEENS

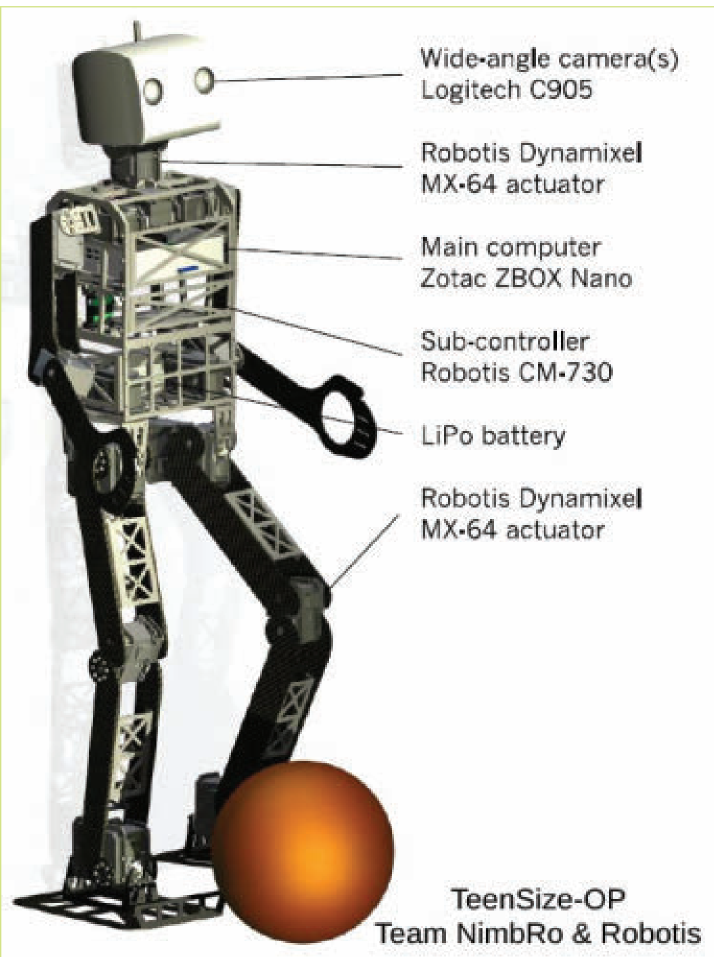


This robot prototype called Copedo was developed at the Universität Bonn, Institute for Computer Science VI, Autonomous Intelligent Systems with the support of the RoboCup Federation and the Korean company Robotis to promote the TeenSize class of the Humanoid League. It is equipped with six DOF per leg, three DOF per arm, and has a two DOF neck. The main computer has a dual-core processor. One of two wide-angle cameras can be placed in the head.

Copedo is 114 cm tall and weighs about 8 kg. Its first competition was RoboCup 2012 in Mexico, where it (with Dynaped) won the TeenSize soccer tournament, the technical challenges, and the Louis Vuitton Best Humanoid Award.

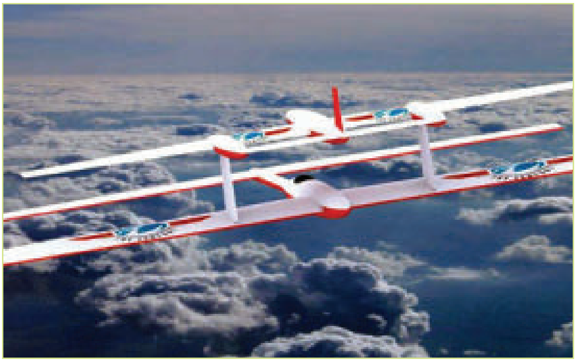
The TeenSize League's minimum height requirement has gradually risen over the years to 90 cm (just shy of three feet). Building a robot that can reliably walk, kick, and stand up from a fall at this height is more challenging than it sounds; only five teams qualified to compete this year.

The TeenSize-OP has a total of 20 degrees of freedom, powered by Robotis Dynamixel actuators. Each leg has six MX-106s; each arm has three MX-64s; and the neck has two MX-64s. It runs the ROS middleware on a ZBOX Nano (1.6 GHz dual-core AMD Fusion processor with 4 GB RAM, SSD, WLAN, USB 3.0, and HDMI ports) and a Robotis CM-730 subcontroller powered by LiPo batteries. It also features two Logitech C905 cameras with wide-angle lenses.



With the help of the community, Team NimbRo (**www.nimbro.net/Humanoid/robots.html**) would like to build modules for visual perception (of the game situation), robot state estimation, inverse kinematics, omnidirectional walking, motion generation, basic soccer skills, robot communication, and game control (by the referee box).

Cool tidbits herein provided by www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, http://www.robots-dreams.com/, and other places.



FLIGHT OF THE CENTURY

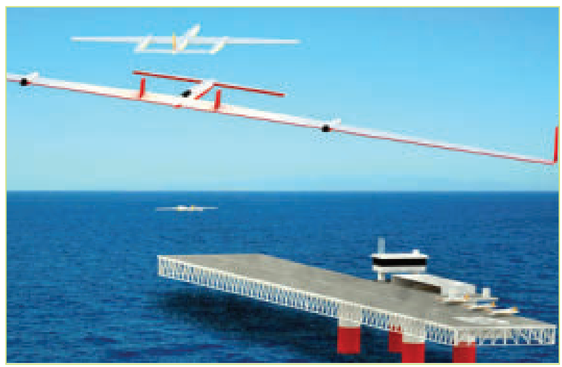
It's hard to beat the energy density of gasoline. You have to go with either compressed hydrogen, something nuclear, or antimatter. This is bad news for everything that runs on electricity which includes all of our gadgets, electric cars, and (more recently) electric aircraft. In order to make electric aircraft viable, a creative solution is necessary, and it doesn't get much more creative than autonomous midair recharging from giant flying UAV battery packs.

The real problem with batteries is that they aren't fuel. They *store* fuel in the form of electrons, but electrons don't weigh anything. With gasoline, it magically vanishes into dirty chemicals as soon as you use it, meaning that your vehicle gets lighter and more efficient as it goes. Batteries, on the other hand, become increasingly more useless as you suck the juice out of them to the point where you're lugging around giant boxes of metal for no reason.

Chip Yates (a world-record motorcycle racer) and a team of engineers think that this is silly, so they've come up with a better idea. Or actually, two better ideas to make electric aircraft more viable, and enable a non-stop flight from New York to Paris that they're calling the Flight of the Century.

Better Idea #1: Jettisoning used battery packs. There's absolutely no reason to carry around the extra weight of a depleted battery with you, but you can't just drop them out of the bottom of your electric airplane. Or, can you? The team plans on rigging its battery packs up with GPS-guided parachutes, and when the packs are depleted they'll be dropped one by one, recovered on the ground, and then recharged and used again.

Better Idea #2: Midair recharging with UAVs. Aircraft that run on gasoline can refuel from flying tankers, so why can't aircraft that run on electricity refuel from flying battery packs? The Flight of the Century team is designing battery-laden UAVs that can autonomously dock with electric aircraft, transfer energy, and then drop away to return to base for a recharge. Over a long flight, an aircraft could take advantage of as many of these UAVs as it needs to keep going. For its NYC to Paris attempt, the Flight of the Century team plans to use five of them, based along the route all the way from Newfoundland to Ireland. It may even be possible to keep an electric aircraft flying indefinitely, using a continuous loop of UAVs that take turns delivering power and recharging themselves on the ground or on marine platforms.



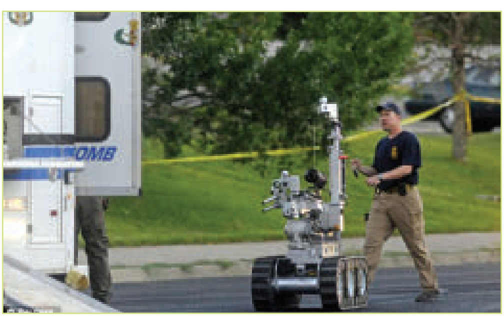
As cool as this system is, it's not going to take the place of jet fuel anytime soon. Chip Yates explains why:

"In the short term, electric airplanes are feasible for specific missions but not as a direct replacement for all fossil fuel burning aircraft. When quiet operations are required or when the military demands a low heat signature for stealthy operation, or for areas with severe noise restrictions or for training aircraft doing many landings and take-offs close to an airport, missions like this the electric plane makes sense. One day if society runs low on fossil fuels or when fuel becomes significantly more expensive, only then can you make a direct cost comparison with electric aircraft."

That day might still be a ways away, but it's important to be thinking ahead and coming up with innovative (and slightly crazy) methods of making renewable energy do what we need it to do. And just to be clear, this whole Flight of the Century thing isn't just a concept. The team is planning battery jettison tests this year, with a transatlantic UAV-recharging flight in 2014.

IN THE NEWS

A robot was sent in to James Holmes' (the suspect in the recent Colorado movie theater shootings) apartment by the bomb squad that placed a "water shot" — a device that emits liquid and shock wave — near the main explosive device. When set off, this successfully deactivated it. FBI Lab experts determined that a trip wire would have been used to mix two liquids that would be set off when the door was opened.



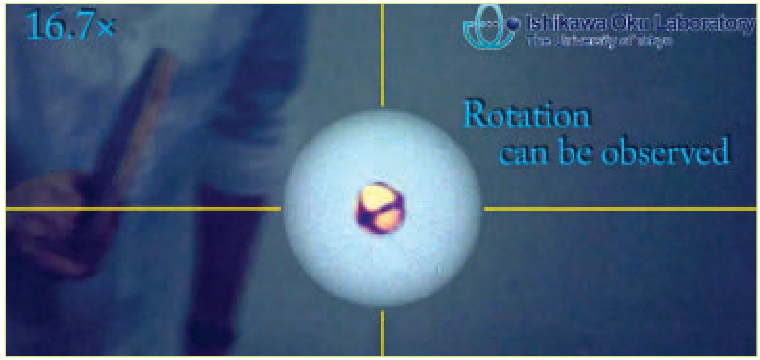
A robot with a camera was used to check out the suspect's car and further study chemicals, aerial shells, and other objects that could detonate or burn in the apartment before humans stepped in. More than 100 bomb technicians, chemists, ATF agents, local police, and firefighters have been working on the case.

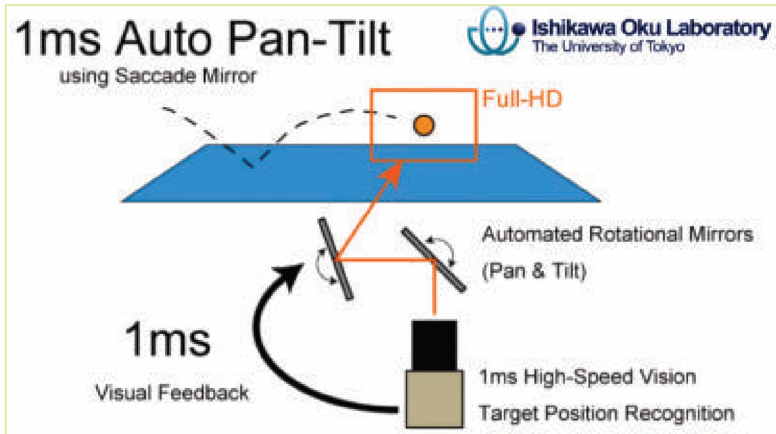
PAN-TILT HANDLING

Professor Masatoshi Ishikawa from the University of Tokyo is showing off their 1,000-frames-per-second camera using a pan-tilt system to track a ping pong ball. The device is so fast it can always keep the ball in the center of the frame.

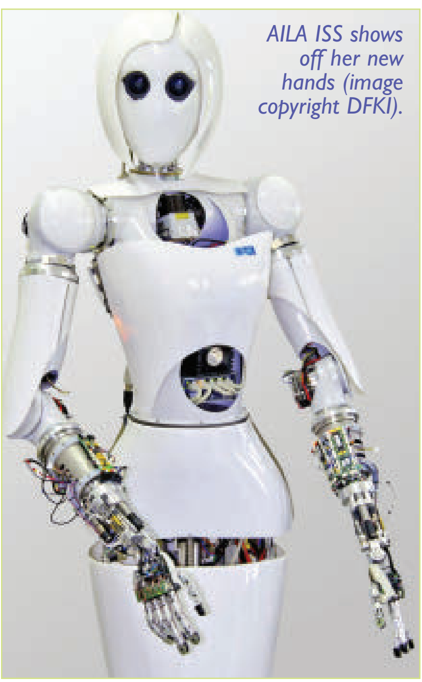
Possible applications include tracking balls or players on sports broadcasts, and recording detailed dynamics of a flying bird or fast moving vehicles.

How do they do it? The camera uses a custom vision chip that monitors what pixels are changing, and by doing that one thousand times per second it can keep track of fast moving objects (bouncing balls, flipping pages, falling eggs, etc.).





Broadcasting sport games is quite popular. Hence, high quality and powerful videos are in great demand. However, it is often hard for camera operators to keep tracking their camera's direction on a dynamic object such as a particular player, a ball, and so on. Current methods have been limited to either moving the camera's gaze slowly with a wide angle of view, or controlling the gaze based on a prediction. Super slow and close-up videos of the player or ball are thought to be especially quite valuable. However, camera operators have not been able to do that as well as they'd like to. To help solve this issue, the Ishikawa Oku Laboratory developed 1 ms auto pan-tilt technology. This technology can automatically control the camera's pan-tilt angles to keep an object always at the center, just like autofocus keeps an object in focus. Even a high speed object like a bouncing pingpong ball in play can be tracked at the center due to a high speed optical gaze controller Saccade mirror and 1,000 fps high speed vision. The Saccade mirror controls a camera's gazing direction not by moving the camera itself but by rotating two-axis small galvanometer mirrors. It controls the gaze by 60 deg — the widest angle — for both pan and tilt. Steering the gaze by 40 deg takes only 3.5 ms. The newest prototype system accesses a full HD image quality for actual broadcasting.



AILA ISS shows off her new hands (image copyright DFKI).

AILA — BESMAN FOR THE JOB

DFKI Bremen's humanoid robot AILA is being readied for work in space, thanks to 3.8 million euros in funding by the German Aerospace Center (DLR). Project BesMan (Behaviors for Mobile Manipulation) will run the next four years to develop the control software necessary to teleoperate robots in space. Specifically, the robot will mimic human movements of the torso, arms, and hands. Already, AILA has been given a new pair of five-fingered hands which are much more capable than the fingerless pads it had before (they only picked up boxes, which doesn't really require fingers). Like NASA's Robonaut R2 and Russia's SAR-400, AILA ISS will be required to grasp and use tools, as well as operate control panels. Although it will be teleoperated by a human on Earth most of the time, it will also need to perceive changes in the environment and act independently should the need arise.

Researchers are already thinking beyond the space station; the software will be designed to work with robots of varying shapes, from humanoids like DLR's Justin to multi-legged climbing robots. It could then be used to teleoperate robots designed to assemble solar panel energy stations on the Moon ahead of a manned mission.

In order to recreate human-like movements, the researchers are experimenting with a motion-capture system. Basically, a researcher in the lab performs an action which is then simulated on the computer. The software will break up the movements into smaller segments that can be sent into space and used when necessary. "We must build systems that approach the capabilities of people," says Prof. Dr. Frank Kirchner, Director of DFKI Robotics Innovation Center and the Robotics Group at the University of Bremen.

COMBAT ZONE

Featured This Month:

www.servomagazine.com/index.php?/magazine/article/september2012_CombatZone

BUILD REPORT:

Siafu: An Army of Ants – Part 4

● by Pete Smith

In this final part of my Army of Ants series, I will describe how I made the drum for Siafu and completed the rest of the bot build.

The drum is made from 7075 aluminum. It's a very tough but still easily machined alloy and ideal for this application. I started by machining the outer diameter (**Figure 1**), and then used a 1/2" drill and a boring bit (**Figure 2**) to machine out a recess to mount the motor.

When I neared the

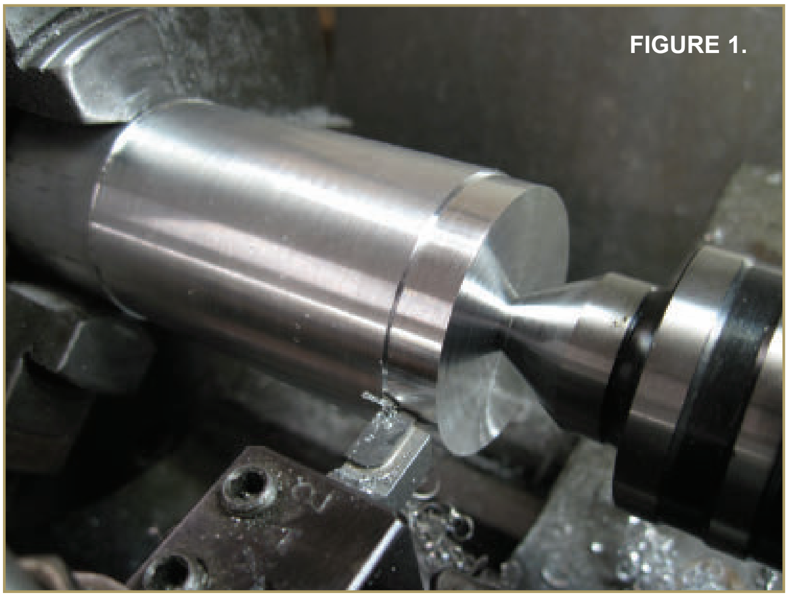


FIGURE 1.

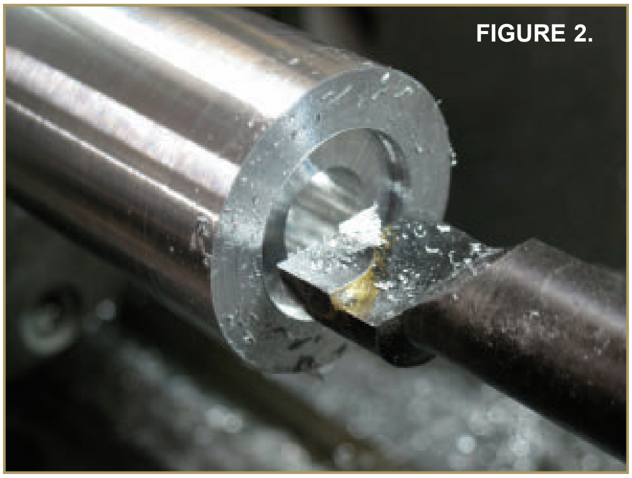


FIGURE 2.

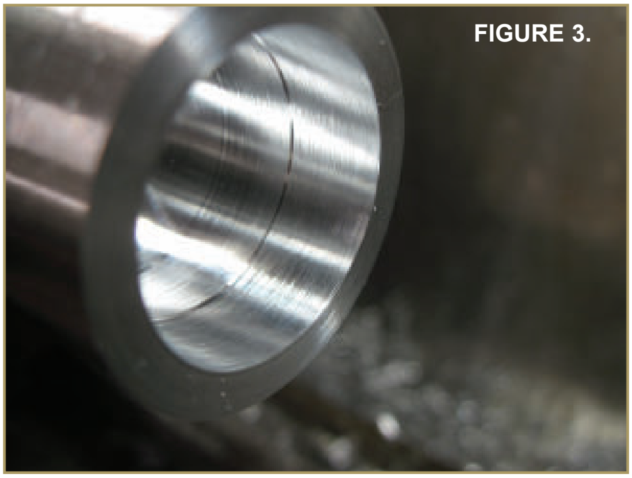


FIGURE 3.

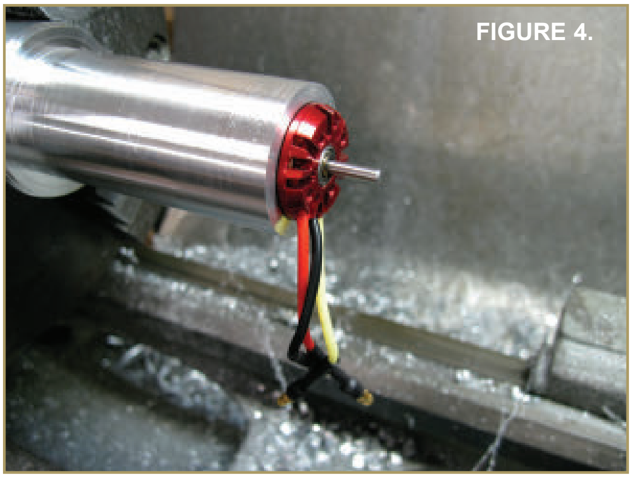


FIGURE 4.

FIGURE 5.

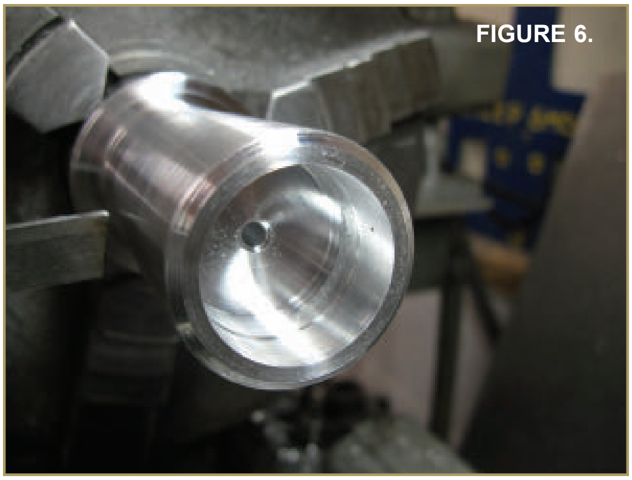


FIGURE 6.

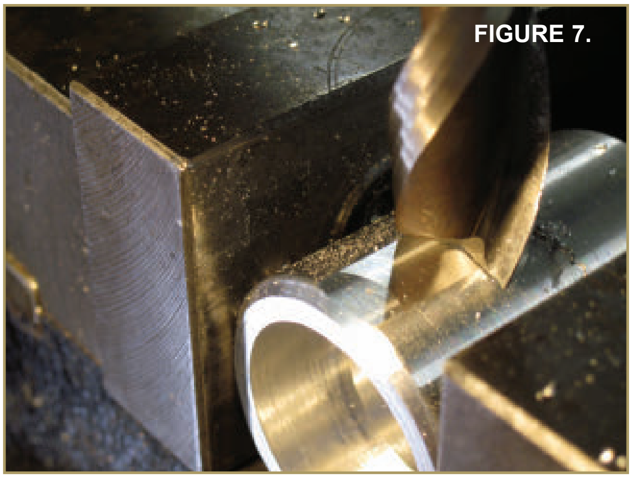


FIGURE 7.

dimension of the motor's diameter, I produced a step in the bore (**Figure 3**) to form a seat for the outer section of the motor. When that section of the bore is a close sliding fit for the motor, the exact depth can be set as in **Figure 4**.

I added a chamfer on the outer edge to ensure it will safely clear the motor's wires, and then center-drilled the hole (**Figures 5** and **6**) for the fixed axle that will protrude from the other side of the drum. I then cut the part off to the correct length.

In order to ensure the tooth mounting holes are opposite each other, I first machined a 0.020" deep flat (**Figure 7**) and then turned the part over so that it rests on that flat. A second 0.020"

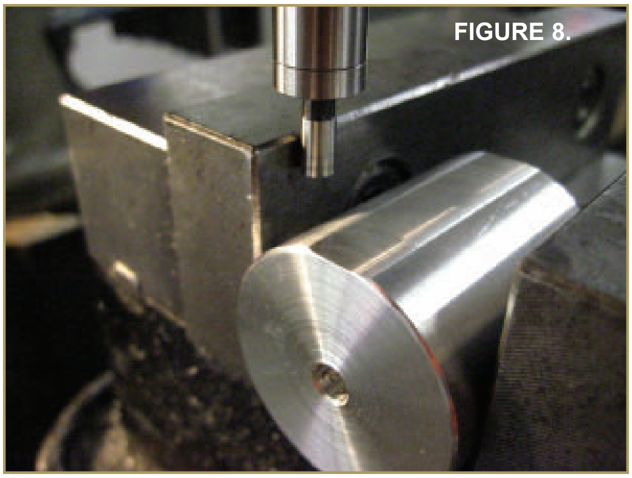


FIGURE 8.

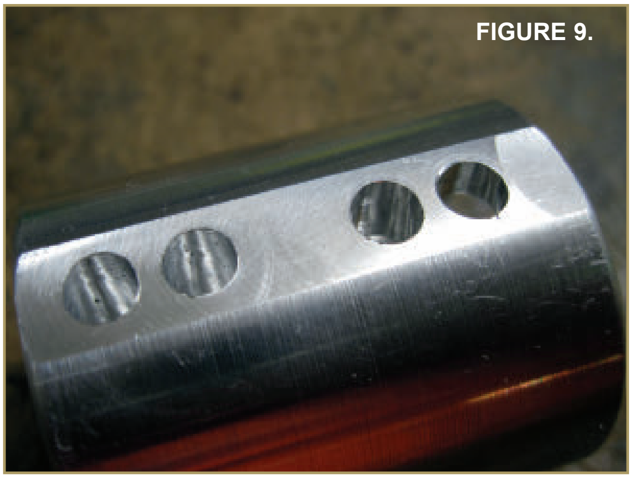


FIGURE 9.

FIGURE 10.

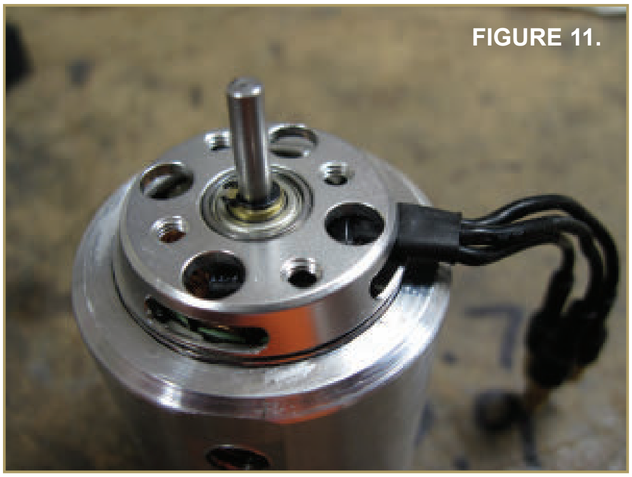


FIGURE 11.

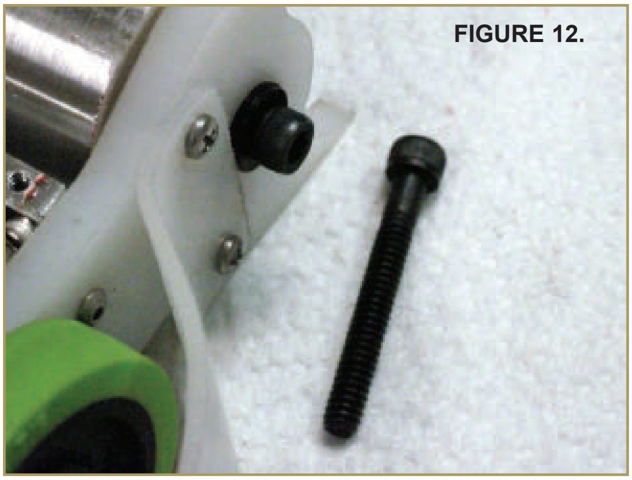


FIGURE 12.

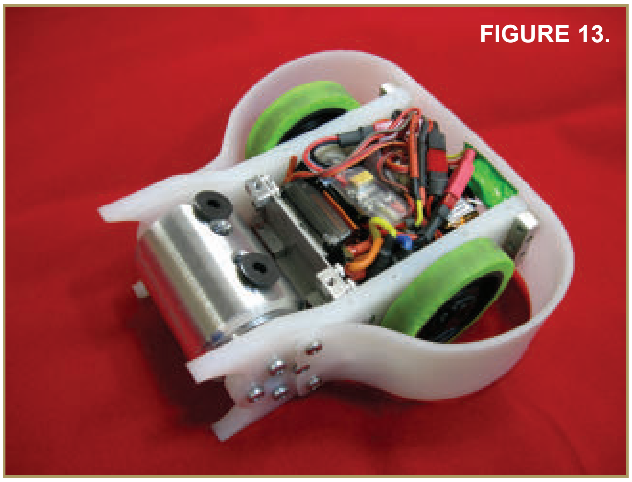


FIGURE 13.

flat is then machined on the opposite side.

An edge finder was used (**Figure 8**) to accurately position the drill for the four holes on each side (**Figure 9**) that will be tapped to mount the teeth.

Once the teeth mounting holes were tapped 1/4-20 (the blind holes require the use of both standard and bottoming taps) and the axle mounting hole tapped with a 10-24 thread, I roughened the surface of the motor (**Figure 10**) with a Dremel, then glued the motor in place using a thin layer of Shoo Goo. I made sure that the motor was sitting squarely on the

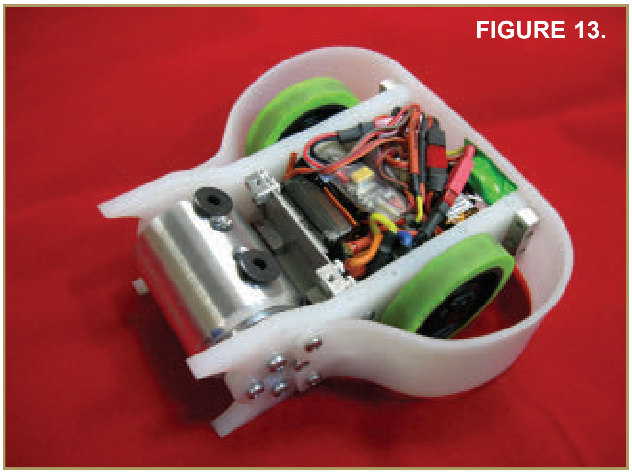


FIGURE 13.

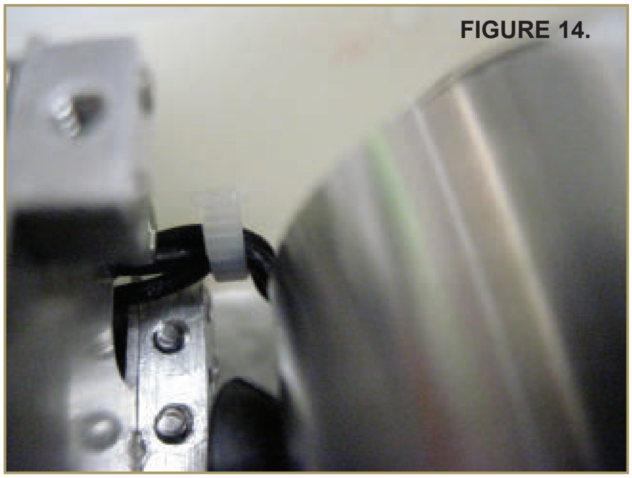


FIGURE 14.

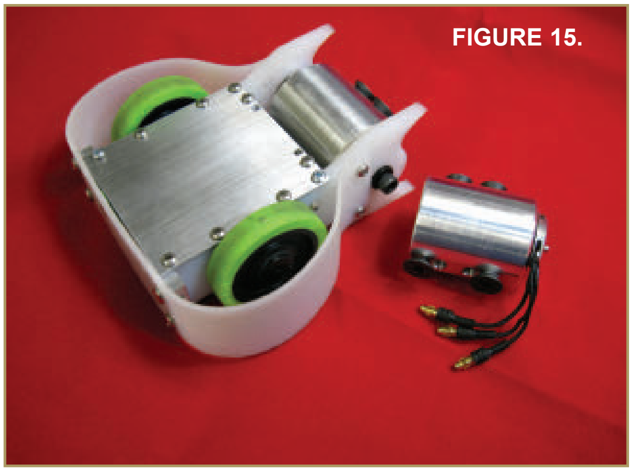


FIGURE 15.

step and then let the glue set (**Figure 11**).

The original intent was to use a shoulder screw as the axle on the other end of drum. However, I made an error in the first drum I built, and drilled and tapped with the wrong thread (10-24 rather than 8-32). I recovered from the error by using a partially threaded 10-24 bolt (**Figure 12**) where the unthreaded section of the bolt is coincidentally the same diameter as the shoulder of the original designed screw. Team Pneusance's Poco Tambor uses a similar solution.

I cut off the excess shaft on the motor, added (use loctite) the 1/4-20 flat head screws as teeth, then fitted the drum and installed the weapon and speed controllers into the chassis (**Figure 13**).

The wiring is a tight fit. The cables for the motor could rub against the drum, so they are held clear by a small tiewrap (**Figure 14**), utilizing slots provided in the chassis wall.

The completed bot and one of its spare drum assemblies can be seen in **Figure 15**. There are some balance issues with the first drum I built, but I hope that those can be resolved with modifications to that drum or one of the others planned as spares. Having fully assembled spare drums is required as replacing a weapon motor during an event would be impractical.

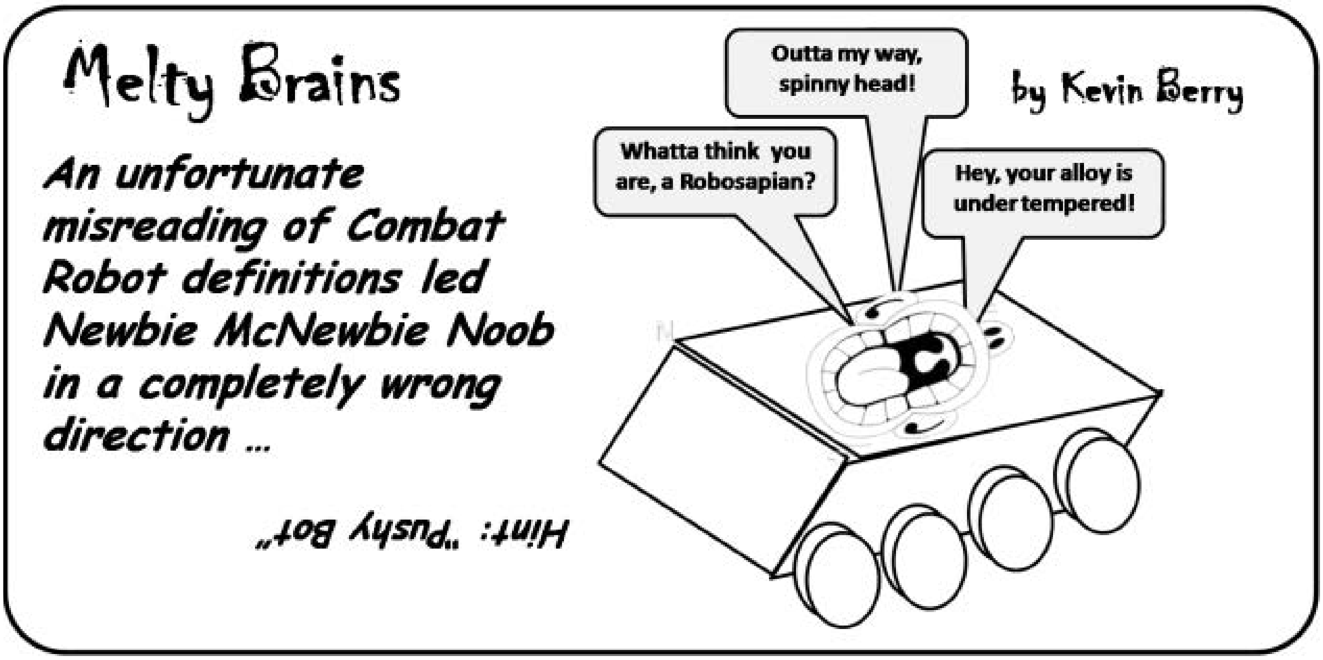
Testing did reveal that the drum can easily run for more than three minutes on the small 2S 300 mAh LiPo battery used. I also plan to try out a 3S battery of a similar size if I can get the drum operating smoothly enough, as the added KE in the weapon and the added speed of the bot could prove useful.

Saifu competed in its first event on July 14th at the Schiele Museum in Gastonia, NC. Also taking part was Klazo, another bot built on the same kit chassis by Mike Jeffries. This could be a great start to a small army of Ants. **SV**

EVENTS

Australian Robowars Nationals 2012 will be presented by the Queensland Robotic Sports Club in Brisbane, Queensland, Australia, September 29th through October 1st. Go to **www.robowars.org/forum**.

Mecha-Mayhem 2012 will be presented by the Chicago Robotic Combat Association in Cleveland, OH on October 13th through October 14th. Go to **www.thecrca.org**. **SV**


Melty Brains
by Kevin Berry
An unfortunate misreading of Combat Robot definitions led Newbie McNewbie Noob in a completely wrong direction ...
Whatta think you are, a Robosapian?
Outta my way, spinny head!
Hey, your alloy is under tempered!
Hint: "Pushy Bot."

MATE ROV 2012 International Competition

by Morgan Berry

www.servomagazine.com/index.php?/magazine/article/september2012_MBerry

Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

In June, students came to Orlando, FL to participate in a competition that encourages students to enter STEM careers. It is put on by MATE (Marine Advanced Technology Education) and represents "a national partnership of educational institutions and organizations working to improve marine technical education in the US." The organization's goal is to prepare students for marine fields. It was founded — along with 10 other Advanced Technology Education Centers — with funding from the National Science Foundation.

Among MATE's other efforts to encourage interest in marine technology careers is the ROV competition which draws hundreds of students from across the globe to compete. The student teams must build an ROV (remotely operated vehicle) that performs underwater tasks. The MATE ROV competition is a unique opportunity for students of all ages who are interested in robotics. The obvious added difficulty of the MATE ROV competition over other student robotics events is that these robots must be completely waterproof. The students also face a series of imaginary scenarios that add real world value to the experience. This year, they had to explore a model ship wreck and fix an oil leak.

An ROV is one of the most important devices in underwater exploration. These tethered underwater robots are used in numerous marine-based fields. Typically, they are controlled by a joystick by workers on the surface. An "umbilical" provides the electricity to the robot. ROVs have at least four onboard engines: two to control the depth and two to control the direction of the robot.

According to the information provided on the MATE ROV competition's website (**www.materover.org**), there are five classes of these robots. Class I are Observation ROVs. They are fitted with cameras, lights, and potentially

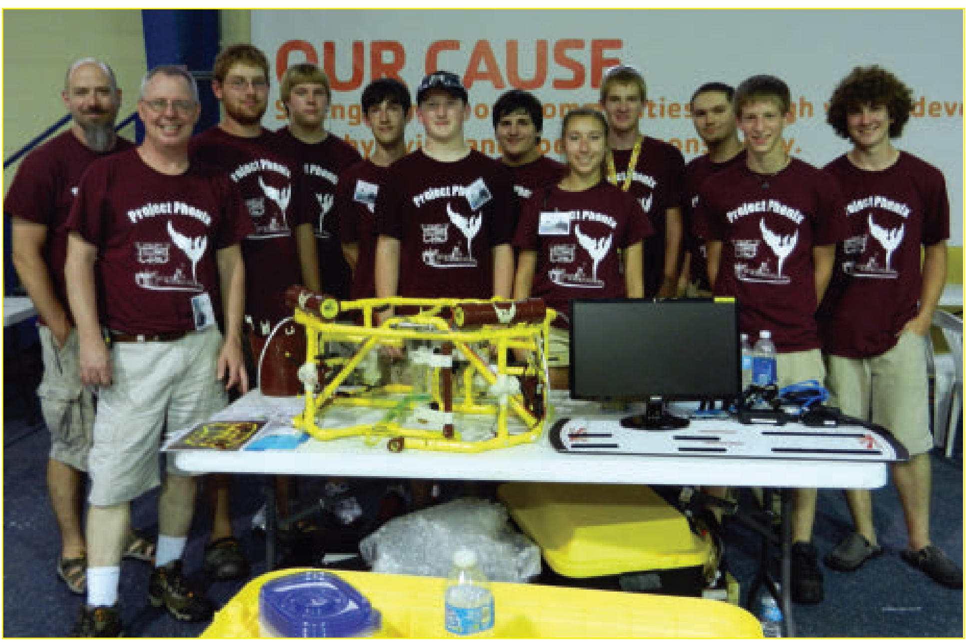
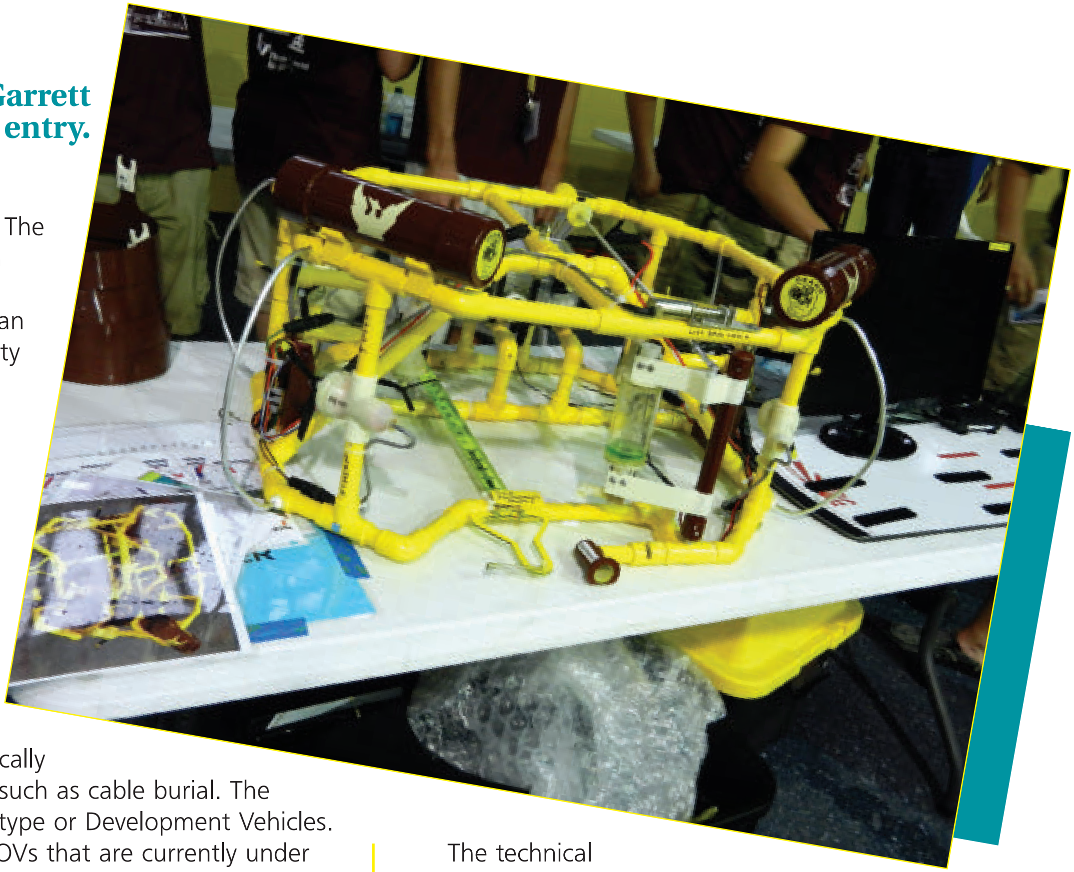


FIGURE 1. ROV team from Garrett County, MD.

FIGURE 2. Garrett County's entry.



sonar. They are intended for observational purposes only. The second class are Observation ROVs with a Payload Option. They are Class I robots with an added small payload capability and/or a basic manipulator. Class III are Work-Class Vehicles. They carry additional sensors or manipulators and are larger and more capable than Classes I and II. Class IV vehicles are Towed and Bottom-Crawling Vehicles. This type of ROV is pulled by a surface craft or winch with limited or no self-propulsion. They are typically designed for a specific task, such as cable burial. The fifth and final type are Prototype or Development Vehicles. This category includes any ROVs that are currently under development, including AUVs (Autonomous Underwater Vehicles).

The MATE ROV competition consists of underwater mission tasks, technical reports, engineering presentations, and a poster display. It is divided into two categories. The high school and beginning college level category is the Ranger division. The upper level category is the Explorer division. There are added tasks that the teams in the Explorer category must complete.

This year, the underwater mission was based on exploring a shipwreck. The teams had to explore the wreck, collect information about it, and then remove fuel from the ship. This teaches students how to design and operate ROVs, but also incorporates real world skills that are crucial in marine technology careers.

The technical report includes design and technical information, cost, and other details about the ROV. The students also had to incorporate business and entrepreneurial skills in the paper by structuring it as a work proposal which is a common task on the business side of

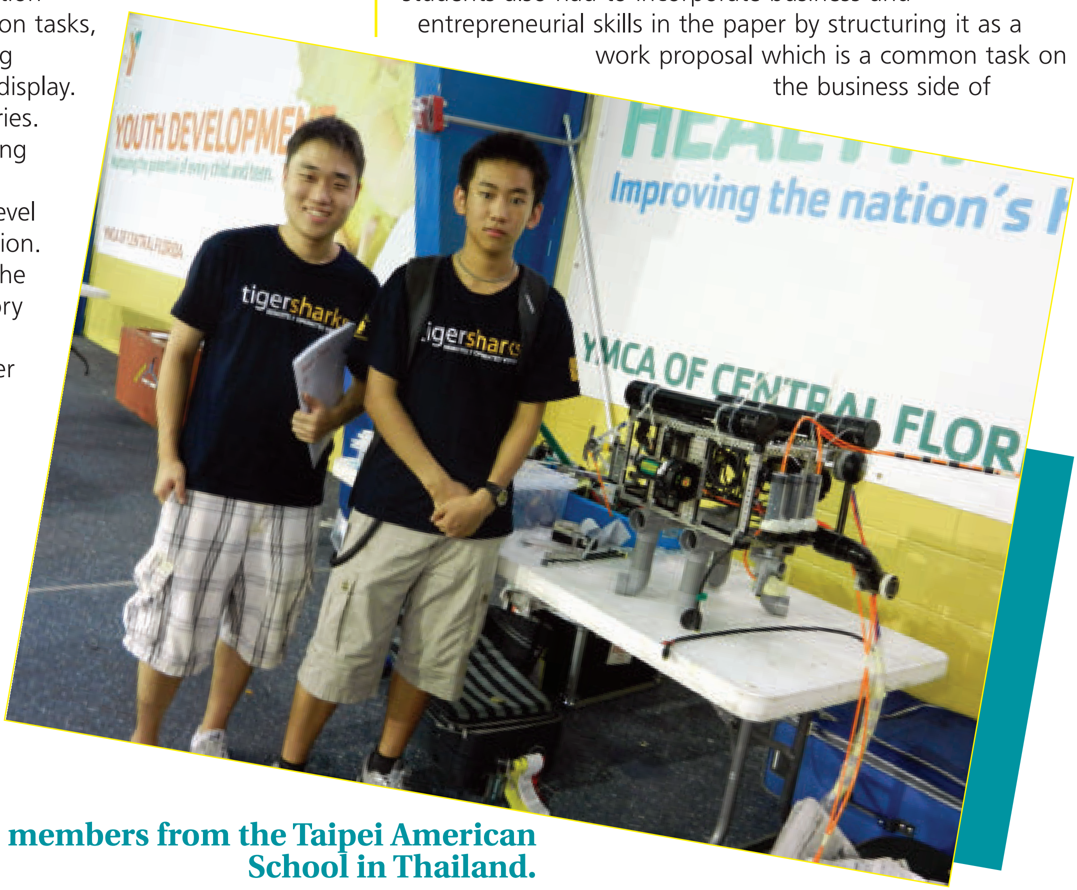


FIGURE 3. Two team members from the Taipei American School in Thailand.

FIGURE 4. Taipei American School's machine.

technology careers.

The engineering presentation is given to a professional panel of judges who work in marine technology careers. After giving an overview of their robot, there is also a question and answer session. The poster includes information about the robot and the business information about their team.

This year in the competition, there were 56 teams from nine countries. Each group had a unique story about their participation in the event.

One team was from Garrett County in Maryland. They competed in the Ranger class. Although this was their first year competing, the team was fortunate enough to have a team mentor who has worked on ROVs in the past, including work on a 1998 exploration of the Titanic. Levi Lantz, the team captain, explained that their goal in the competition was speed and agility. When building their robot, they worked as a team to build the main frame of the vehicle, and then split off into smaller groups to focus on completing each piece of the mission. Although they were new to the MATE ROV competition, this was by no means their first experience with student robotics matches. Many of the team members had participated in FIRST Robotics, as well as the FIRST Tech Challenge.

Another team traveled significantly farther than Maryland — all the way from Taipei City in Taiwan. This team was comprised of students from the Taipei American School — a private school in Taiwan's capital city with an American style curriculum. The school was founded in the 1950s for children of American personnel serving in Taiwan. Because they are an international team, members Kevin Ku and Anthony Lin explained there is the added challenge of shipping their robot across the world.

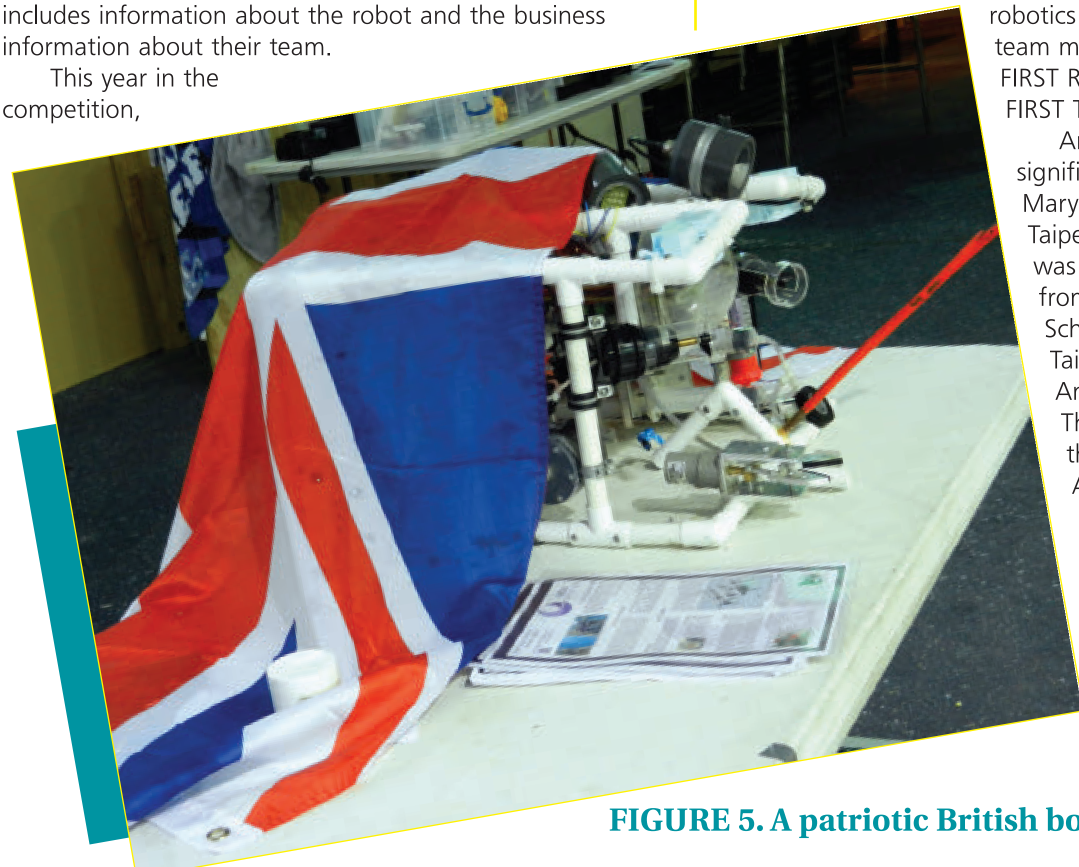


FIGURE 5. A patriotic British bot.

As opposed to local teams, international teams like this must spend

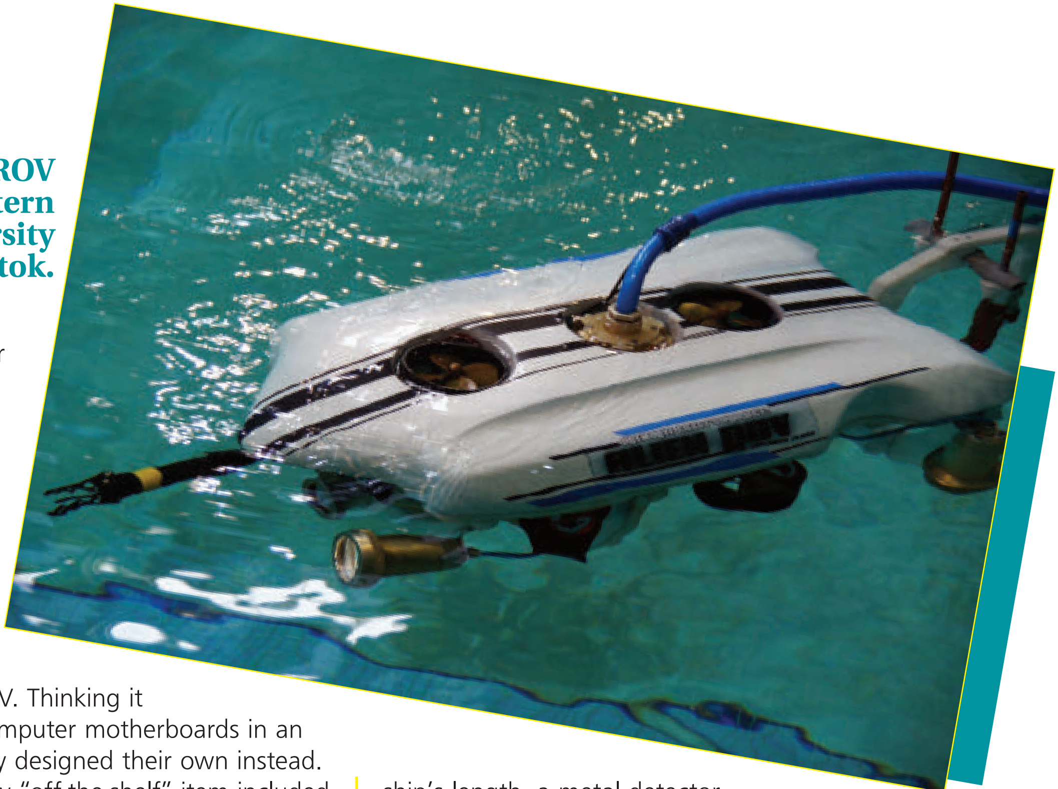


FIGURE 6. ALIEN ROV from the Far Eastern Federal University in Vladivostok.

part of their time during the tournament reassembling their ROV. Like other participants, this team had a "divide and conquer" strategy; they split up into small groups to handle a portion of the robot building. They wanted to combine as many functions as possible into as little hardware as necessary. Because of this, the team custom-built much of their ROV. Thinking it inefficient to include entire computer motherboards in an entry like some teams do, they designed their own instead. In fact, the motor was the only "off-the-shelf" item included in their robot. Everything else was custom-built by the team. They were fortunate enough to have a supportive school, who paid for the robot as well as the travel expenses to the bout.

The winners of the Explorer's division came from the Far Eastern Federal University in Vladivostok, Russia. The team attributes their success to the variety of technical knowledge in their group. There was a combination of five different specialties in technological fields.

Because the Russian team could not find many products locally, the time it took to order parts became a challenge. They had less time to develop and troubleshoot their ROV because they often had to wait on parts to be delivered from Moscow. However, since this was the fifth year the team competed in the MATE ROV event, they were able to adapt to these obstacles and build a first place robot. The bot itself was designed with six thrusters, four video cameras, "a tank flushing device, measuring tape for determination of the ship's length, a metal detector to search for the debris of the vessel, a holder for the magnetic patches, and a manipulator for attaching the light bag to the mast and transporting coral." (You can read more about this ROV in Kevin Berry's article starting on page 40.) All of this cost $10,420, which was paid for by the university and the Administration of Vladivostok. The travel costs set the team back another $26,377, which illustrates just how difficult it can be for international teams to compete in events such as these. SV

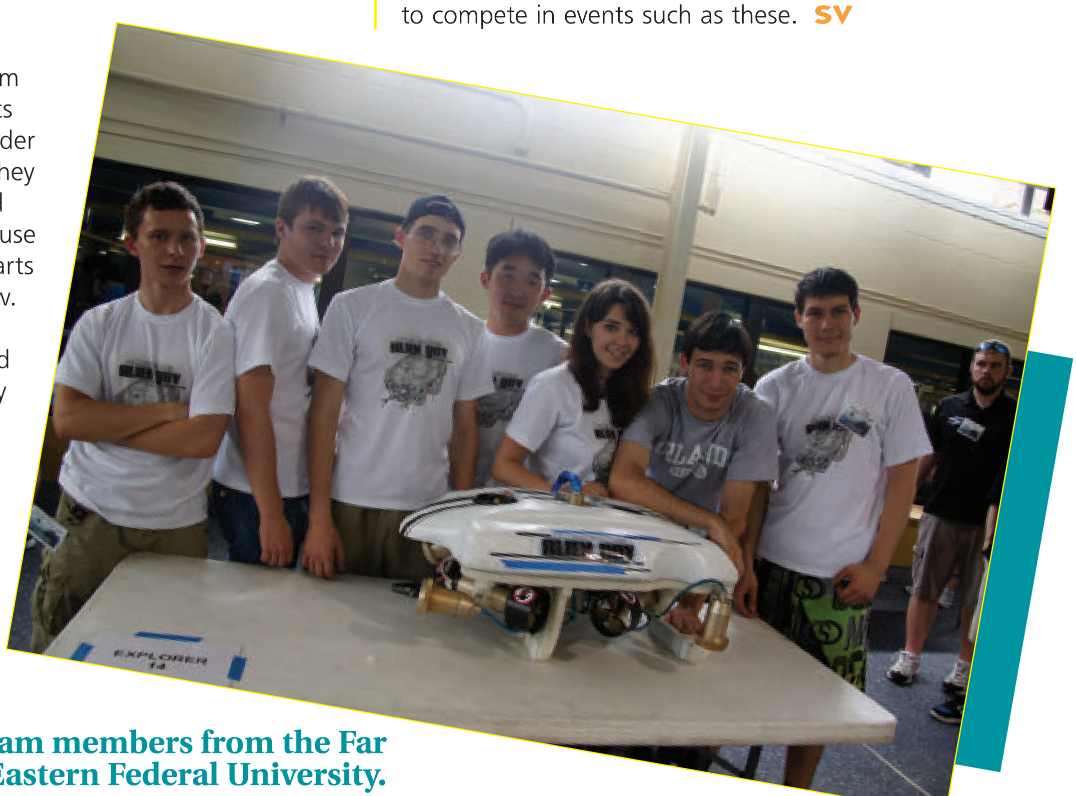


FIGURE 7. Team members from the Far Eastern Federal University.

Far Eastern Federal University's ALIEN ROV Revealed!

by Kevin Berry

The Far Eastern Federal University (FEFU) student team won the world championship in remotely operated vehicles at the recent 2012 MATE International ROV competition which was held in Orlando, FL.

The FEFU team — which goes by the name Primorye Coast — competed with more than 20 teams from different countries including the USA, China, India, Great Britain, Egypt, and others. FEFU students have been taking part in these competitions since 2008. The team won first place for the first time in 2010. Primorye Coast consists of students with different specialties — from computer security to medical physics and interior design.

This year, the theme of the competition was the research of sunken World War II vessels. For the participants, there was a simulated event in which the fuel materials of the sunken vessels were an environmental threat that had to be neutralized. The teams were challenged with the development of methods for a secure fuel extraction.

The "Primorye Coast" company prepared a technical paper as part of the required submittal for the competition. This article summarizes that paper, with some editorial comments and rewrites to make the (over) 5,000 word document flow in this condensed format.

The ROV has six powerful thrusters that can provide a steady position while working with a variety of devices, or making video captures with four cameras. A special payload was installed on the vehicle, including a tank flushing device, a measuring tape for the determination of the ship's length, a metal detector to search for the debris of the vessel, a holder for the magnetic patches, and a manipulator for attaching the lift bag to the mast and also transplanting corals.

Total expense for the development of the vehicle was $10,420, while the total project cost — taking into account the materials and the cost of travel — was $40,133.

Initial Design Concepts

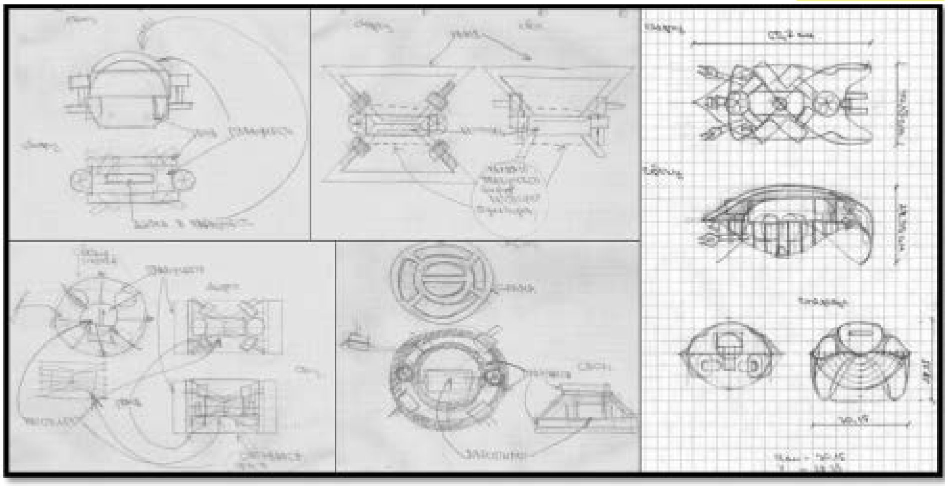
While developing the vehicle, the team decided to create something new — something not like their previous vehicles or industrial models, but capable to perform mission tasks.

They began with a classic brainstorming session. All team members became vehicle designers for a while. Team members proposed dozens of ideas and sketches (**Figure 1**). Sometimes heated discussions were held. "Will it float at all?" "How would it stand on the ground?" "Should the center of masses really be here?" and other more specialized questions were asked. The two best designs were simulated with SolidWorks. The final choice is shown in **Figure 2**. The team notes that at this point, the most difficult task was still ahead: proving their design to their teachers and mentors. This design — a new concept unlike others the teachers were familiar with — had to be shown to be suitable for mission tasks. The mentors actually did not accept the design at first. After more dynamic discussions, all parties came to a compromise in details (but defended their initial concept).

Frame

The basis of the vehicle's design is the frame (**Figure 3**).

FIGURE 1.



It was designed on the principles of bionics, and therefore resembles the skeleton of a marine animal. The frame is made of polypropylene which was chosen because of its durability and because its density is less than the density of water — it grants the ROV additional buoyancy. The frame consists of an upper plate with the majority of devices attached to it and five "ribs" required for the payload installation.

There is a buoyancy mass made of polyurethane foam mounted on top of the upper plate. The shape of the buoyancy mass is noteworthy. For cutting the buoyancy mass from polyurethane foam and making the necessary forms, the team used a home-built "burning hot string" with a flowing direct current of 4A. **Figure 4** shows team members cleaning up ribs after the initial cuts.

Propulsion System

The ALIEN ROV has six thrusters. Four horizontal thrusters are located at 45 degrees to the longitudinal and transversal axis of the vehicle and provide movement and stabilization in the horizontal plane: lag, run, and course. The horizontal thrusters are attached with special clamps to the upper plate of the frame that has special openings for the clamps. These clamps were also designed in SolidWorks, as well as the frame. Two vertical thrusters provide movement and stabilization of depth and pitch. **Figure 5** is a collage of thruster components. These consist of the motor, propeller, and thruster control units (TCUs).

They used Faulhaber 4490 H 048BS-K312 motors. This decision was justified by good characteristics such as the absence of magnetic losses, low power consumption, and compact size. Engine power is also great: 212W at 10,000 rpm. They designed propellers themselves, especially for these motors. They were made of plastic using a 3D printer.

There is an integrated TCU in the housing of each thruster. TCUs and the control board are communicating via a CAN interface on a global bus. In addition, it monitors current consumption and temperature to prevent overcurrent and overheating.

Housings

Containers for the electronic components were designed to withstand the pressure at a depth of six meters. The housings are made of aluminum, with O-rings for sealing the containers. The containers consist of a cylindrical housing — a chassis to host the electronics boards and two caps (**Figure 6**). The top cap is used for an input tether and output wires to the thrusters. The bottom cap has the output for the wires from the cameras, lights, manipulator, pressure sensor, and metal detector.

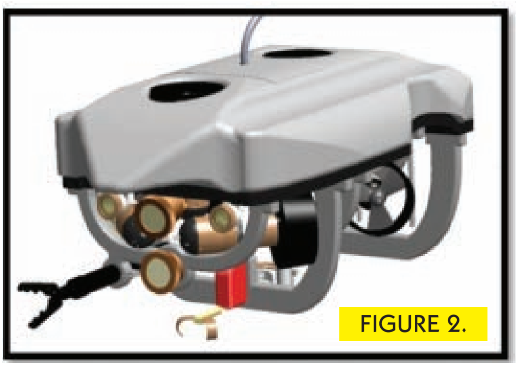


FIGURE 2.

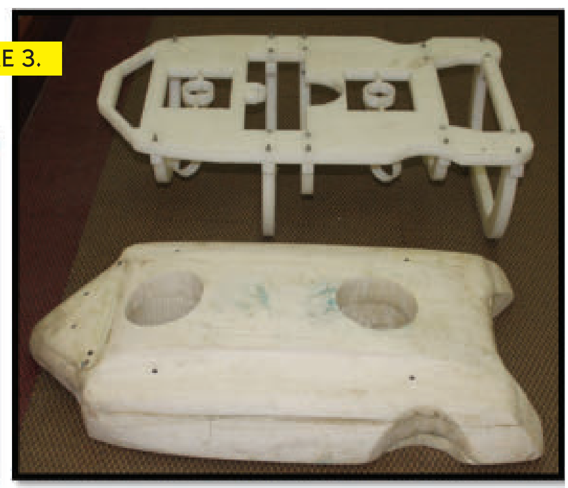
Safety Features

Safety is very important to the Primorye Coast team. There are safety features in the vehicle and safety rules that every member of the team must follow. The ROVs safety features include:

- Thermal sensors in thrusters.
- Leak sensors in cameras and electronic unit container.
- Electric fuse.
- A kill switch for emergency power-off.
- Shrouded propellers to prevent injury.
- Warning signs.

Their safety rules consist of two parts. The first part regulates safety during constructing and maintaining the

FIGURE 3.



vehicle; their second part established rules for deployment, operating, and transporting the ALIEN ROV.

Electronics

"The main part of the vehicle's systems is an electronic unit. Like a human heart, it drives all other units and systems to precisely perform the orders of the pilot," the team notes.Control signals to the electronics components are passed through the controller board which is based on an STMicroelectronics (**www.st.com**) TE-STM32F207 microcontroller. It controls pressure sensors (Honeywell ASDX015A24R-DO, and SensorsOne DMP 331; **www.sensorsone.co.uk**), a SparkFun electronic compass (HMC6352), a CRS accelerometer, and a manipulator control board; it also controls voltage to the decoder of a multiplexer. A video multiplexer receives signals from the four available cameras to transfer all video streams to the surface, where a demultiplexer provides the desired video stream to the operator's monitor. A power board supplies all the electronics components with necessary voltage (24V, 12V, 5V). Also, the microcontroller board supplies 3.3V that is necessary for some of the components.

Cameras and Lights

ALIEN uses VM32HQ-B36 modular color cameras from Video Security (**www.videosecurity.ru**). They chose these due to their small size, ease of installation, high sensitivity (0.1 lx), and availability. Among the other advantages of the camera is backlight compensation — which is useful for underwater observing — because the observed objects will be placed on a background of bright light.

The basis of lights is a SibProekt "Photon" MR-0209 torch (**http://sibopt.ru**) which has nine LEDs in a waterproof housing. An unnecessary container for batteries was removed since the lights are connected to the control board power supply.

Tether

The data cable is designed to transmit signals between the electronics unit and the switching block. For video transmission, the team used two 1.5 mm coaxial cables with an impedance of 75 ohms. The ROV requires rather high power, so they chose two 6 mm power cables. To transmit control signals, a twisted pair cable is used.

To collect all the wires in one tether, the team creatively bought a rubber garden hose which was used as a sheath. Broaching wire in the hose is very laborious, but rather exciting work. They ended the process of broaching at the top of a five-story building, stretching the tether to the entire height of the building and straightening all the wires inside.

Commutation Unit

The commutation unit is used to separate the tether into different lines: power, control, and video.

The 48V from a power supply comes to the commutation unit through the fuse which protects the circuit against excess current. The power is then supplied to the board of power switches and after that, to the ROV. The video signal passes through the video modulator and then appears as part of the GUI on the laptop screen. Control signals are transmitted between the operator console on the surface and the electronic unit onboard via an Ethernet interface. An AC/DC converter was integrated to the commutation unit to power the vehicle from AC power.

Software

Primorye Coast stuck with open source software. Selecting the most up-to-date open source development tools and free operating system, they built the control board on Ubuntu OS using the Qt Source Development Kit and open source libraries. All communications with the ROV are made through the Ethernet interface that is compatible with Linux OS. Also, Ethernet was the most appropriate solution with the best combination of reliability, quality, speed, and availability.

FIGURE 4.

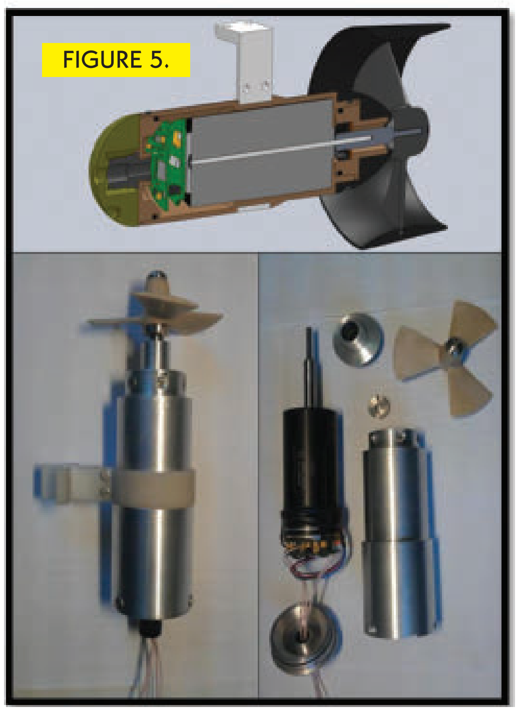


FIGURE 5.

Control Board's GUI

The GUI (**Figure 7**) has a modular structure. Modularity allowed the programmers to vary appearances and split the whole programming task into small problems. Each developer worked out only his own problems to reduce the number of conflicting code changes. The developed widgets can be easily used in other projects because all the source code is publicly available. Another advantage is that widgets can be separately used for the ROV's systems debugging. The GUI consists of several widgets: depth, roll-pitch, joystick, manipulator control, LEDs, cameras, and others. Important information is always clearly visible for the pilot. The ROV is controlled mainly by joystick but the pilot can use the keyboard, as well. This provides an ability to have two pilots working cooperatively.

The main window periodically calls functions which provide a data exchange between the ROV and widgets, and allocates data for the next processing.

They also provided a possibility to configure TCUs from the surface "on the fly" (or rather, "on the swim"). That gave an ability to easily change engine parameters without reflashing, therefore saving time. All configurations are saved in structured XML documents. It's quite visual, informative, and handy.

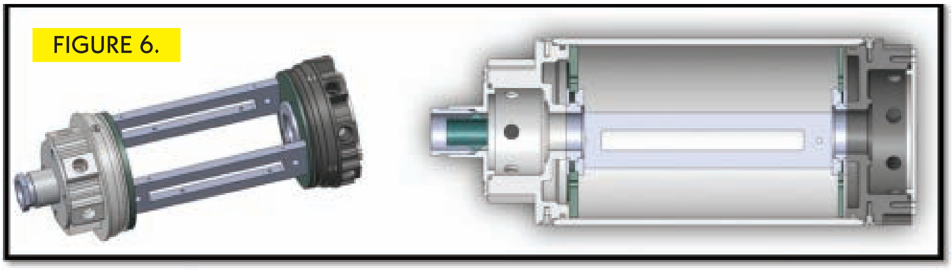
Firmware

The brain of the vehicle is a Terraelektronika TE-STM32F207 board based on the Cortex-M3 microcontroller. Its function is the processing and transfer of data between the electronic components onboard, and providing communication with the control unit on the surface. To ensure the most effective interaction of all systems, they use various interfaces for data transmission onboard: CAN (for thruster control); SPI (for three-axis gyroscope and accelerometer); and I^2C (for gathering data from a digital compass and an internal pressure sensor).

For communication with the operator console, they use Ethernet as mentioned. The board is also programmed to serve as an autopilot, solving an important task of stabilizing the vehicle.

The firmware consists of logically separated parts of program code that perform determined functions (sending packets through the certain protocols, providing PID stabilization, collecting data from the sensors). They run pseudo-parallel due to a specially configured system of hardware and software interrupts with priorities.

FIGURE 6.



PID Controller

Primorye Coast members designed an autopilot built on a microcontroller, allowing the use of autopitch, autohead, and autodepth stabilizations. A quite effective proportional integral derivative (PID) mechanism was implemented for accurate vehicle stabilization. The proportional term produces an output value that is proportional to the difference between the desired and current values. The integral term accelerates the movement of the process towards setpoint and corrects statistical errors.

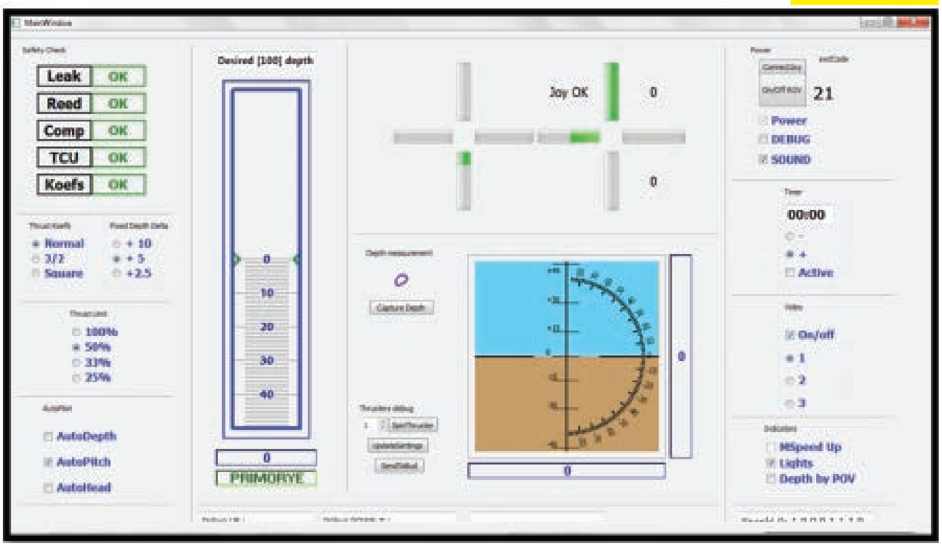
Derivative control is used to reduce the magnitude of the overshoot produced by the integral component and improve the combined controller-process stability. The team paid special attention to configuring the PID controller because roughly chosen parameters could easily cause a destabilization of the vehicle which has to be avoided at all costs in order to successfully complete the mission.

Details of the mission payloads, failures and troubleshooting, future enhancements, team lessons learned, and some rather interesting personal observations are available at **http://dvfu.ru/files/upfiles/documents.may12/FEFU_Technical_Report_2012.pdf**.

The unique educational environment that is being created on Russky Island (Vladivostok) will provide students and the people of Primorsky Krai with ideal opportunities for studying and making the most of their creative potential.

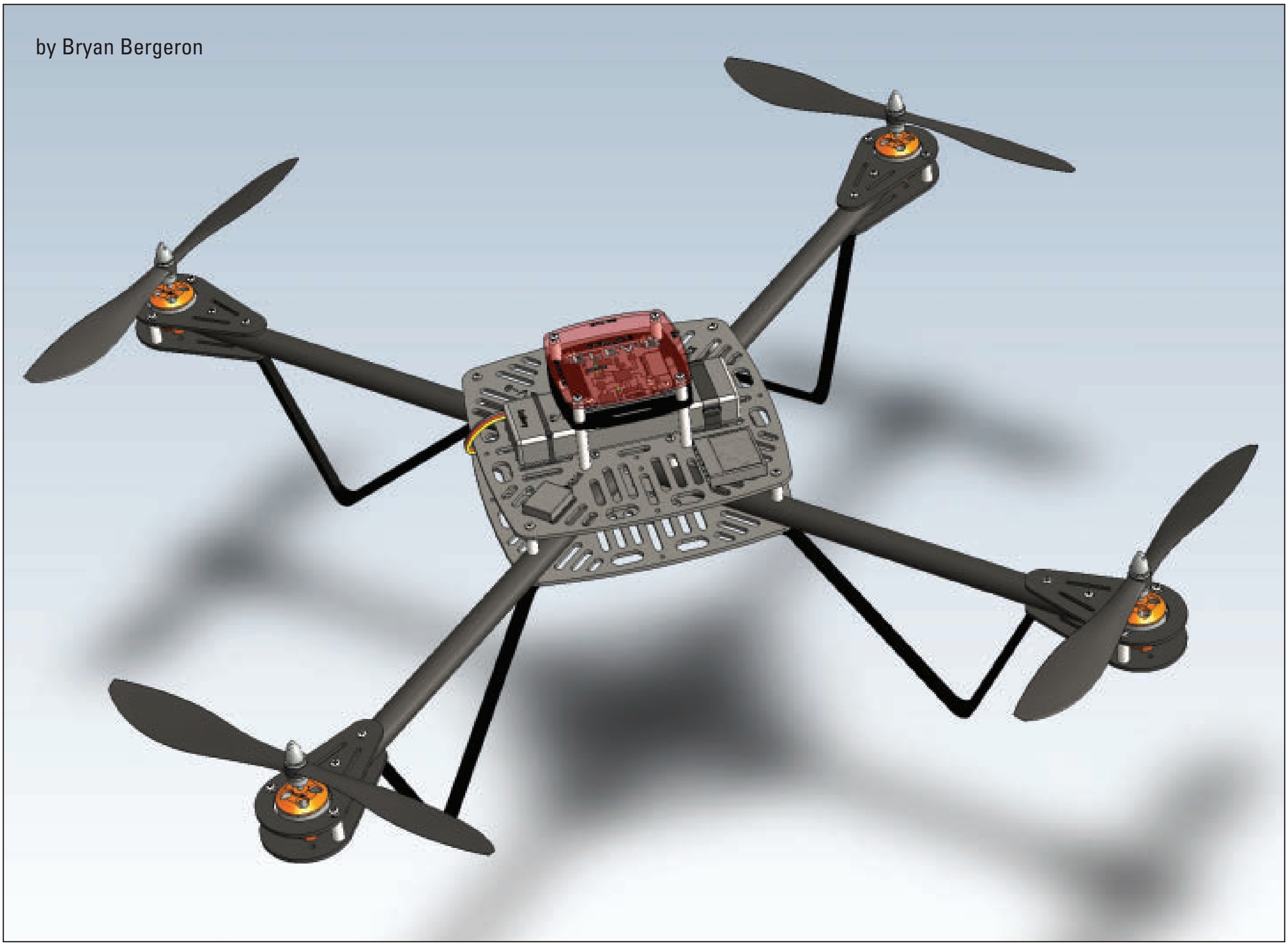
The Marine Advanced Technology Education Center (MATE Center) is the organizer of the MATE ROV Competition 2012 and other international competitions on underwater robotics. **SV**

FIGURE 7.



Parallax Elev-8 Quadcopter — Part 1: The Mechanical Build

by Bryan Bergeron



www.servomagazine.com/index.php?/magazine/article/september2012_Bergeron
Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

Ready to take your robotics experiments to new heights? Follow along as I walk through the Elev-8 Quadcopter kit from Parallax. In this two-part series, I'll assume that you're proficient in robotics and electronics but haven't worked with a flying platform. The goal is to show you what's involved in terms of cost, infrastructure, and knowledge. I'll focus on the mechanical build of the platform in this article, and devote a second article to the electronics and setup.

Introduction

When I was a kid cutting my teeth on .049-powered model planes, I used to long for one of the 'real' model kits — powerful engines, huge wingspans, and speed. When I finally had the money and experience to pick up one of those kits, I discovered there was little in the way of instructions. The kit consisted of exploded 3D diagrams and a lot of uncut balsawood. The manufacturer apparently assumed that anyone buying such a model had graduated past the need for hand-holding. Well, the Parallax Elev-8 is one of those 'real' models.

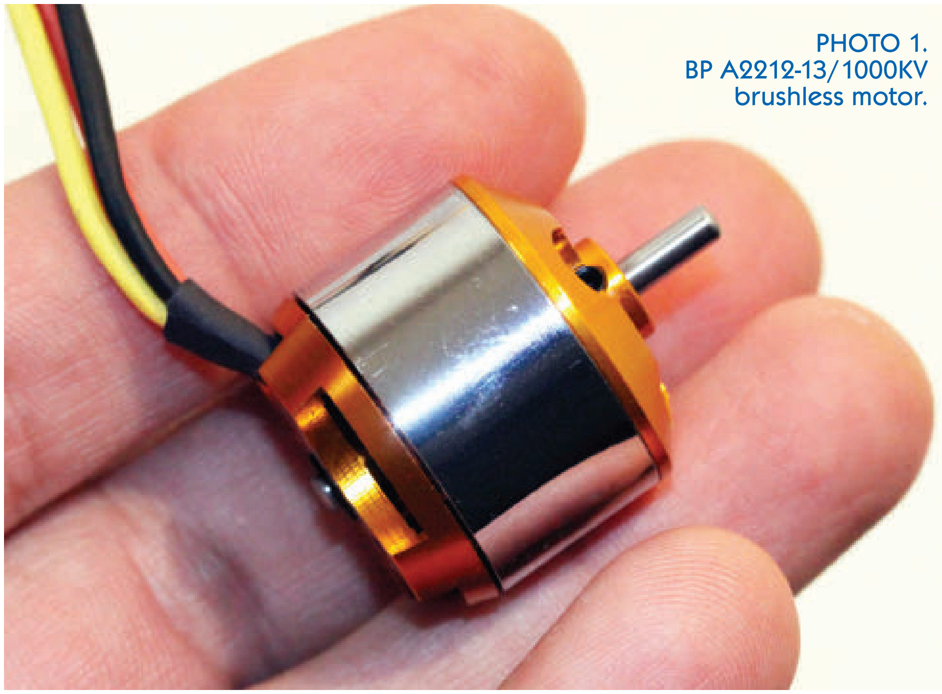


PHOTO 1. BP A2212-13/1000KV brushless motor.

The folks at Parallax offer a flying robotics platform that is wicked fast, super responsive, and yet has enough space and thrust to transport two pounds of gear — anything and everything from cameras and GPS receivers, to paintball guns and egg launchers. It's also a serious experimental robotics platform built with open source hardware and software, and it's fully user customizable. Of course, the flight computer — the HoverFly Open Board — is based on the aptly named Parallax Prop chip.

The Elev-8 is for experienced electronics and model aircraft builders and — when it comes to flying — for experienced R/C enthusiasts. That said, you have to start somewhere. The build is simple enough — if you can put together a carpet crawler, you can build this kit. However, as fair warning, I built two quadcopters from scratch before attempting this kit, and I still managed to make a couple mistakes during the build.

Another factor to consider up front is cost. This is a $1,200+ adventure. The basic kit sells for $599. For that, you get the motors, speed controllers, HoverFly Open Board flight computer, mechanical structure, and propellers. Add another $45 for the Elev-8 Crash Pack (you WILL crash), and $50 for a pair of LiPo batteries (Sky Lipo 4400 mAh, 3s, 11.1V).

If you're starting from R/C ground zero, then you'll also need a six channel R/C transmitter and receiver (Spektrum Dx6i, $220), a good LiPo charger and power supply ($100-200), a few R/C specific test instruments ($50), optional ultrasonic range finder ($30), low battery indicator ($5), prop balancer ($20), and assorted cables and connectors ($30). Add a still or video camera of your choice, GPS receivers, servo-controlled mounts for your camera or laser, or what have you. The point is, it can add up. Of course, you'll need basic electronic construction hand tools, a DMM, temperature-controlled soldering iron, solder, and testing supplies. Lastly, you need space. You'll need a dedicated 3 x 6 work space for at least a week for construction and testing. You'll also need space to fly. Lots of it.

The Build

In the following discussion, I'll hit the high points of the build as I recommend it, as well as issues that may not be obvious to a first-time quad builder. Officially, this is a six to eight hour project, but this assumes no modifications and everything on hand (never the case for my projects).

Unpacking and Parts Identification (20 minutes)

This kit arrives in a small white box, with components nicely packaged in labeled and sealed plastic bags. Clearly, someone at Parallax thought about the builder. After reviewing what's what, separate the small hardware in a muffin tin or small parts container. This is an appropriate time to marvel at the four BP A2212-13/1000 KV brushless motors. Each 28 mm x 28 mm motor (shown in **Photo 1**) weighs only 53 gm.

The other item to note is the small 1" transparent plastic light pipe which I managed to lose within minutes. There are two pairs of safety glasses in the box — put on one pair now.

PHOTO 2. Apply loctite and tighten both setscrews.

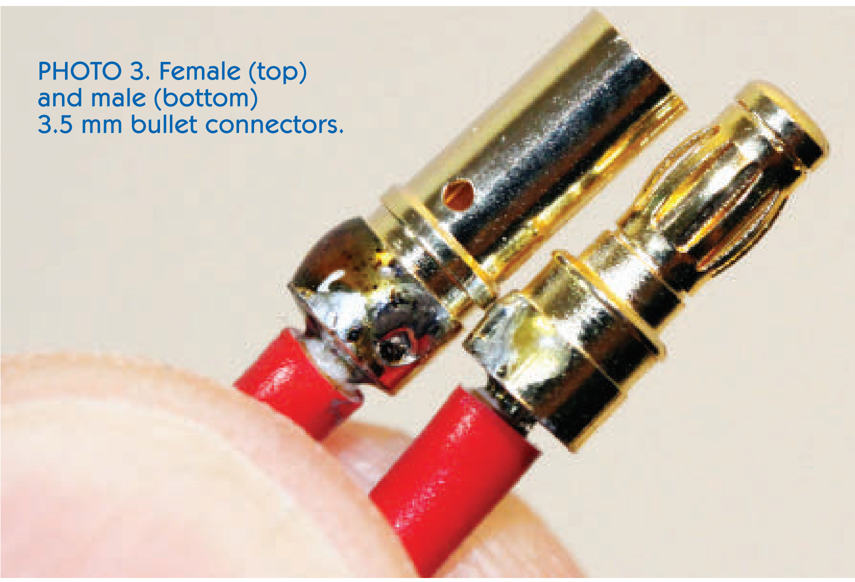


PHOTO 3. Female (top) and male (bottom) 3.5 mm bullet connectors.

PHOTO 4. Joined 3.5 mm bullet connectors.

Motor Prep (20 minutes)

Remove the pair of setscrews from each brushless motor, apply the supplied loctite, and then reinstall the setscrews (see **Photo 2**). If the setscrews loosen in flight, the propeller and half of the engine will separate from the frame, and that's that.

Motor-ESC Wiring (20 minutes)

We need to connect the motor to the electronic speed controllers (ESCs). Start by soldering male 3.5 mm bullet connectors to the three leads from each motor. Next, solder female 3.5 mm bullet connectors to the three blue leads from each ESC. Solder male connectors to the black and red power input wires. By the way, connector sex is important — you want the source to be a protected (with shrink wrap) female connector, as opposed to an exposed male connector.

Photos 3 and **4** show separated and joined bullet connectors prior to adding shrink wrap. Note the male connector is free of solder along the mating surface. The easiest way to ruin a male bullet connector is to allow solder to run down the side of the mating surface which prevents it from compressing during insertion into the mating connector.

Take twelve 12" lengths of the red silicone wire and make extension cables with a male 3.5 mm connector on one end (for an ESC) and a female on the other (for a motor). Use shrink wrap to cover the exposed connection.

You can also solder wire to wire, but then if you ever want to move the motors to another craft or replace a damaged motor, you'll need a soldering iron. You may need to buy extra connectors — check your supply before you begin.

Motor-Boom Assembly (1.5 hours)

At this point, the craft is about to take shape. Using the supplied 4-40 hardware — including nylon spacers — prepare the plastic motor mounts as in **Photo 5**. Create a sandwich with the aluminum tubes as in **Photo 6**. The hardest part of the operation thus far is snaking the motor leads with extensions down each of the four aluminum tubes. Be careful not to pinch the wires when you insert the bolts in each arm. The finished booms are shown in **Photo 7**.

During this step, you can attach the supplied checkered vinyl tape to the arms — red and white for the front two arms and black and white for the rear. You can also attach the two supplied LED strips to the bottom of the forward booms.

I opted against both and simply painted the forward booms and mounts red with Krylon Fusion spray paint.

The undocumented feature that I missed at this step was to first remove the black plastic film from both sides of the plastic motor mounts.

Assembly of Boom Arms to Chassis Plates (20 minutes)

Attach standoffs to the top chassis plate which is identical to the bottom plate. Using standard 4-40 hardware and nylon spacers, attach the arms and feet to the chassis plates. In my model, both red arms face forward — important for flying and for properly orienting the onboard flight computer.

Although not necessary, I also attached the top plate, just to make certain everything fit properly, as shown in **Photos 8** and **9**. Remove the top plate when you're satisfied everything fits as it should.

Layout of ESCs (30 minutes)

The quadcopter is a symmetrical beast, and that symmetry should be reflected in the alignment and positioning of the four ESCs. The goal is to maintain the center of gravity and balance point in the center of the structure. So, treat each of the four triangles formed by the aluminum arms and the bottom plate identically.

I used tie wraps to experiment with different ESC and wiring configurations, and ended up with the ESCs tucked into each apex of their respective triangular spaces, with the ESC flat against the bottom plate (see **Photo 10**). Note shrink wrap is only loosely applied so that the signal connections can be changed later during setup.

Programming and Prep of the ESCs (10 minutes)

Although this is supposed to be a rundown of the physical build, this is an opportune time to program the ESCs. We could wait until later and spend an hour fiddling with the R/C transmitter or use the Turnigy programming card, available from Parallax (see **Photo 11**). The card enables you to program ESCs with parameters such as type of battery, cutoff voltage, timing mode, startup mode, and others that won't make sense until later. All you have to do is set the LEDs on the card to match the table in the HoverFly Open Board manual for each of the four ESCs and you're done. It just takes a few minutes per ESC, but it saves an hour or more of fiddling. More than worth

Photo 5. Motor mount assemblies with motors attached to bases and nylon spacers attached to tops.

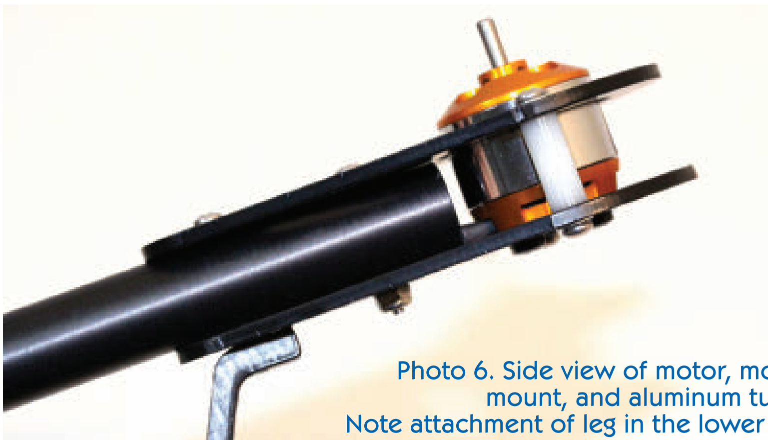


Photo 6. Side view of motor, motor mount, and aluminum tube. Note attachment of leg in the lower left corner of the photo.

PHOTO 7. Completed motor arm assemblies. Red arms are for the front of the craft.

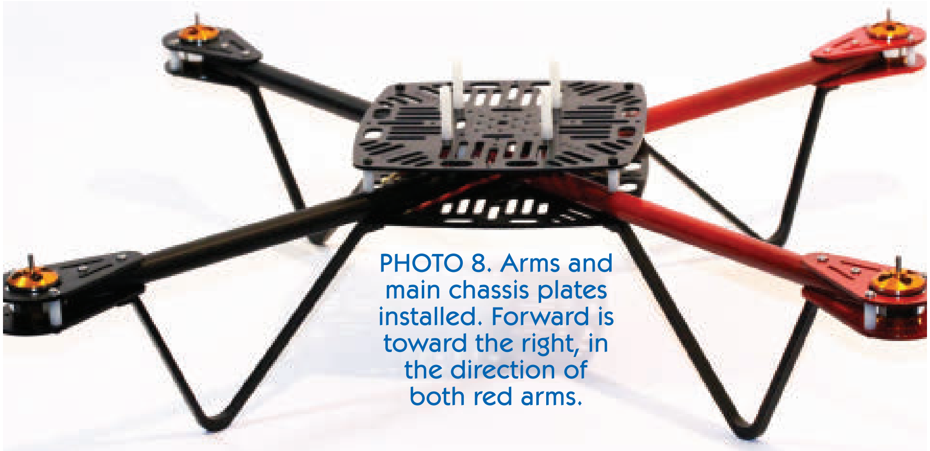


PHOTO 8. Arms and main chassis plates installed. Forward is toward the right, in the direction of both red arms.

PHOTO 9. Close-up of arms and main chassis plates.

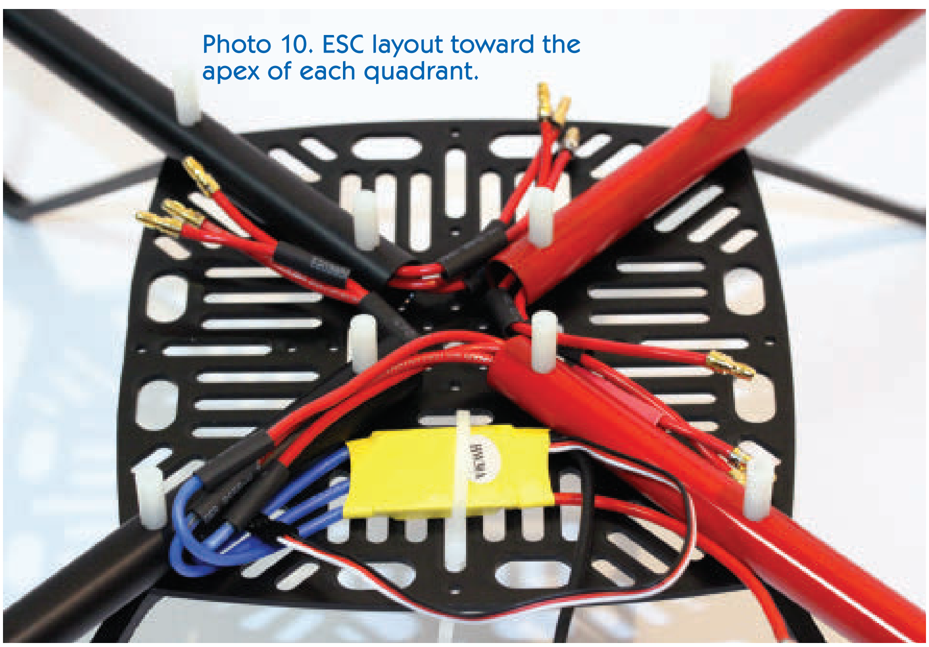


Photo 10. ESC layout toward the apex of each quadrant.

the $10 for the card.

Once you've programmed the ESCs, the next step is to prevent all but one of three ESCs from supplying power to the flight computer. The HoverFly Open Board requires only one +5 VDC at 2A supply, and multiple, parallel supplies can cause problems — especially if one supply generates a significantly higher or lower voltage than the others.

You can remove the +5 VDC output pin from the three-pin connector on all but one of the ESCs. However, you risk permanently damaging the ESC connector.

I prefer to add a 4" servo extender cable to three of the ESC leads and disrupt the power there. On either end of each extender cable, remove the power pin by carefully lifting the plastic retaining tab and then pulling out the wire and pin as in **Photo 12**. Keep the wire intact so that you have the option of using the extender cable later in another configuration.

Power Distribution Bus Fabrication (1.5 hours)

This is the "heavy lifting" part of the build. It's a simple task — supply power from the LiPo battery to each of the ESCs. The challenge is making the connection with minimum weight and resistance.

We're talking about supplying 15+ amps to each of the ESCs at times. Even at idle — without the motors active — the ESCs draw 350 mA at 11.1 VDC. Obviously, there's no power switch in this circuit. There's too much current involved.

I made a power harness from the supplied 12 gauge wire, removing the insulation from the middle of one wire that connects the battery to one of the ESCs. The remaining three power wires attach to the main wire. I used 3.5 mm female bullet connectors on the ESC ends of the harness and a single 4 mm bullet to match the battery connector. Your battery may require a different connector or a different male/female mating combination.

Photo 13 shows the starting

wires and the soldered harness. **Photo 14** shows a close-up of the main solder joint. Given the significant heat and time involved, solder must have wicked at least an inch past the insulation on either end of the joint.

Of course, all exposed 3.5 mm bullets either get covered in shrink wrap or inserted into a connector housing. I prefer shrink wrap for space and weight savings. The downside of shrink wrapping the connection is that replacing or repurposing the ESC requires removing the shrink wrap.

Photo 15 shows the power harness attached to the ESCs and ready to accept the battery connection (far left).

If I had to do it again, I'd probably go with a commercial power bus ($2-$4 at HobbyKing). It's worth doing once, but creating a bus from scratch is a major time sink. However, if you like packing your own chute, then build your own. You never really know how much heat or solder some worker (or robot) in China applied to a connector.

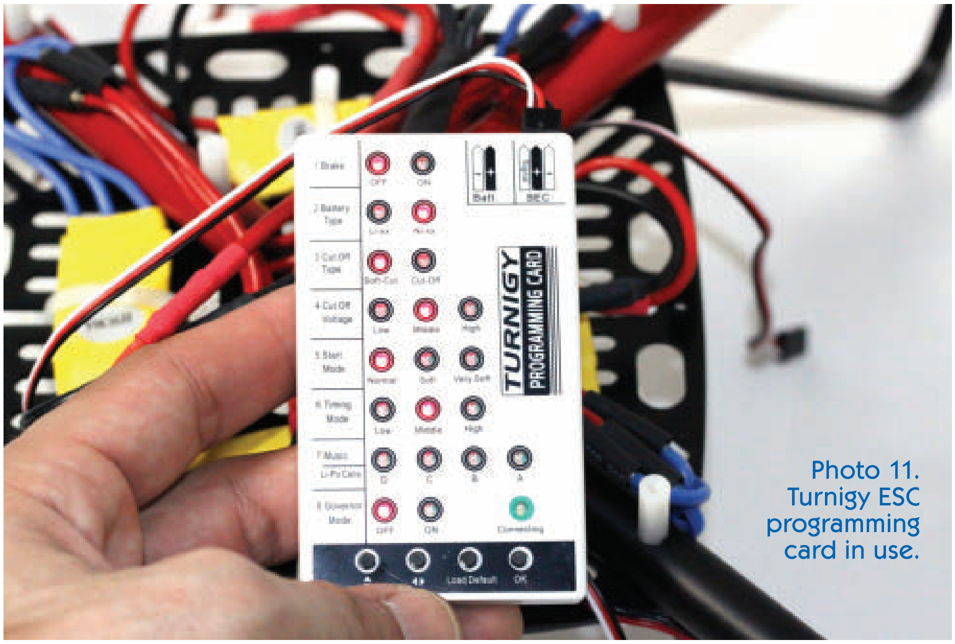


Photo 11. Turnigy ESC programming card in use.

Rotation Check and Rewiring of ESCs (20 minutes)

The quadcopter is able to maintain stability and maneuverability in part because two of the rotors spin clockwise and two spin counterclockwise. Looking down on the quadcopter from above with the two red arms (in my copter) facing away from you, the motors at 2 o'clock and 8 o'clock should be configured to spin clockwise. Configure the other two motors to spin counterclockwise.

Unfortunately, there is no universal color coding for the polarity of motor leads. You'll have to check the rotation of each motor and then — if a correction is needed — switch two of the three ESC leads to reverse the direction of rotation. Fortunately, a little servo/ESC tester can verify proper rotation direction in seconds. Simply plug in the ESC's PWM input plug to the tester and 11.1 VDC to the power input leads.

Any servo tester will do, especially since you just need to get the motor spinning for a fraction of a second. I use the Turnigy Mega Meter ($40, HobbyKing) which you should consider. Alternatively, a dedicated servo tester ($10, TowerHobbies) will do. In a fix, an Arduino set up with the standard servo library is another option. However you manage to verify the motor direction, secure the heat shrink on the ESC connection when you're done.

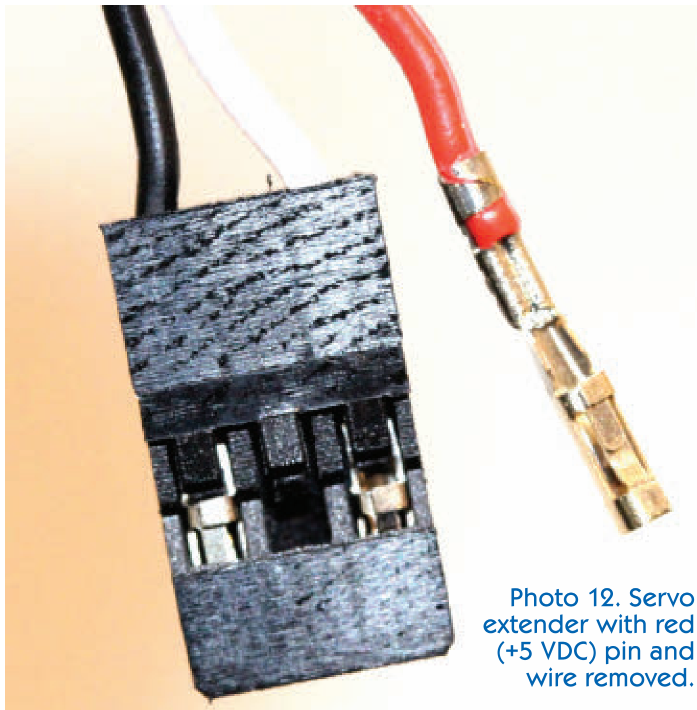


Photo 12. Servo extender with red (+5 VDC) pin and wire removed.

Photo 13. Power harness 12 gauge wires prepped for soldering (top), and then soldered (bottom).

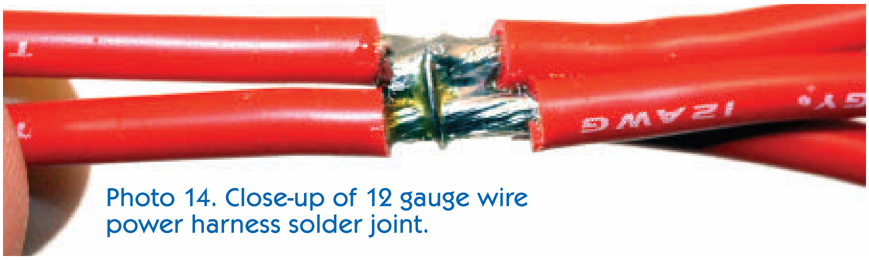


Photo 14. Close-up of 12 gauge wire power harness solder joint.

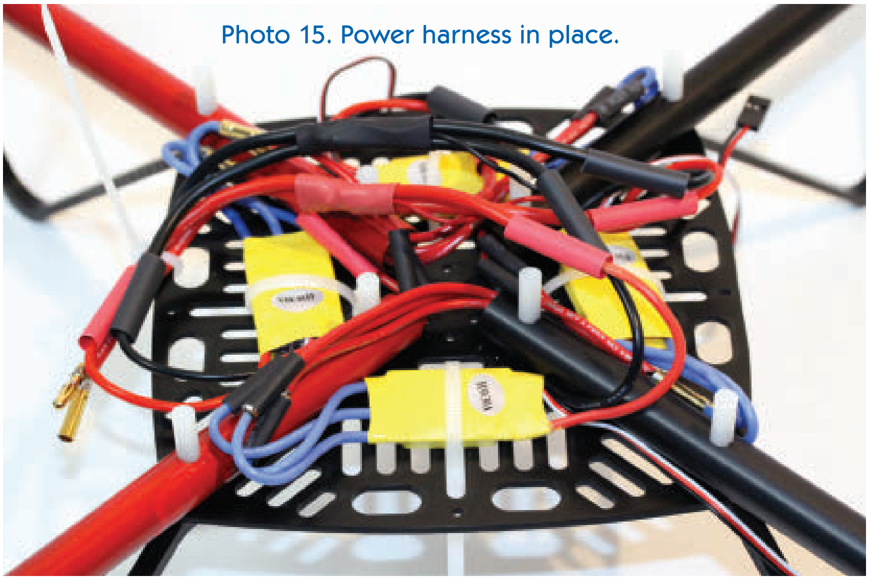


Photo 15. Power harness in place.

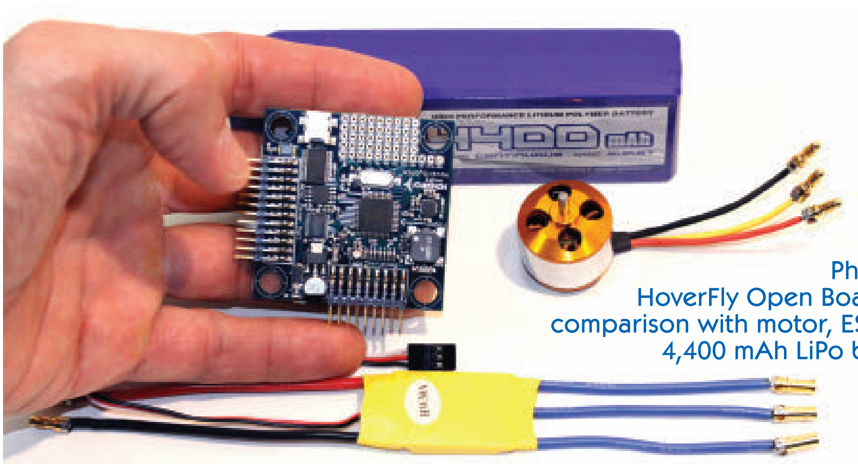


Photo 16. HoverFly Open Board size comparison with motor, ESC, and 4,400 mAh LiPo battery.

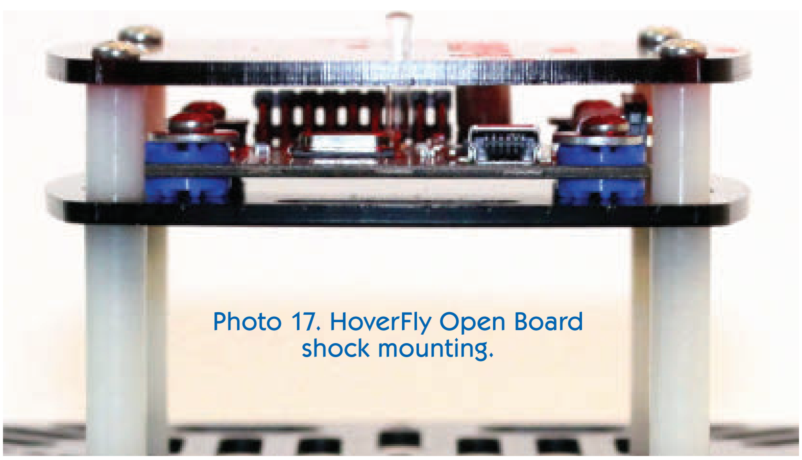


Photo 17. HoverFly Open Board shock mounting.

Temporarily Mount the HoverFly Open Board (10 minutes)

The HoverFly Open Board is not much larger than an ESC (see **Photo 16**). At this point, the instructions suggest mounting the board using silicone grommets and self-tapping screws as in **Photo 17**. However, I managed to strip two of the bolts during wiring and testing. In place of the original metal screws, I installed nylon 4-40 hardware which works fine. I suggest you use tie-wraps to secure the board until you're ready to fly.

Prop Balancing and Test Mounting (1-2 hours)

The final step in the physical build is to balance the propellers, install a shaft adapter, and test the mount. It's the balancing act that takes time, and that's going to make the difference between a flying machine and a self-destructing egg beater.

To balance each prop, you can use a jig and razor or sandpaper or whatever method works for you — just make certain each propeller is balanced or the craft will vibrate wildly. I use a Turnigy R/C Balancer ($13, HobbyKing) and a set of coarse to fine Emory boards.

Next, insert a 5 mm plastic shaft adapter into each of the props. Each prop ships with a set of six adapters — pick the adapters that best fit the aluminum collets; 5 mm worked for me.

Recall that two propellers turn clockwise and two turn counterclockwise. You should be able to tell from **Photo 18** that the pitch is such that downward thrust results from clockwise rotation.

Mount the balanced props with collet adapters onto the motors. Slide the prop onto the adapter, screw on the bullet shaped cone, and tighten using a small hex wrench inserted through the holes in the cone. The cones need to be good and tight.

If there is slippage, now is the time to repair or replace the collets. Remove the props and collets in preparation for electronic setup and testing.

Closing Thoughts

The Parallax Elev-8 is a quality, open platform with an active support forum and quick domestic access to parts and accessories. The price may seem a little steep at $599, but considering that the HoverFly Open Board alone sells for $120, it's reasonably priced.

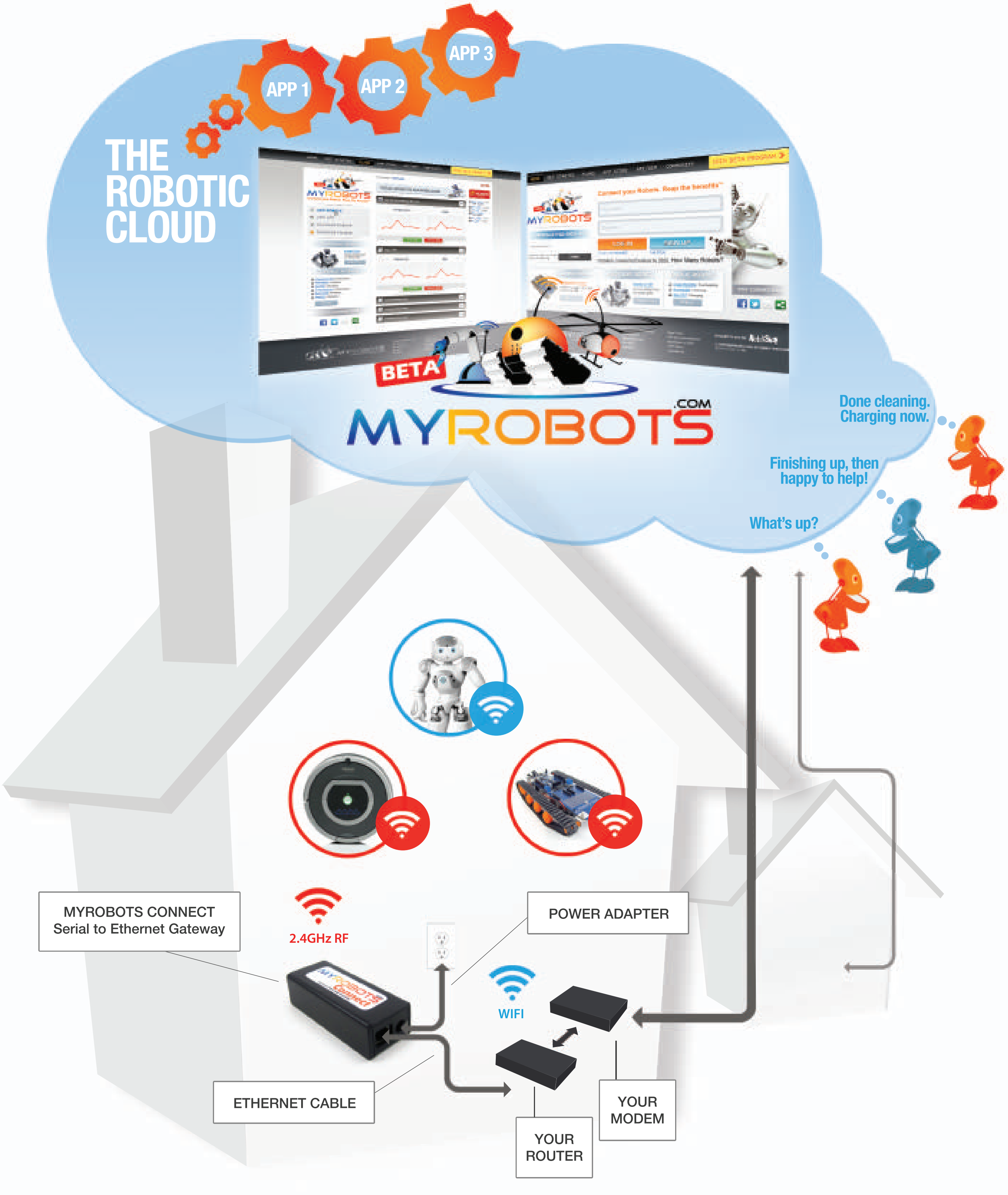
In addition to the large payload capacity, one of the selling points of the Elev-8 is access to parts for repair or modification. Bend an aluminum arm or lose a 4-40 bolt, and simply pick up a replacement at your local hardware store.

Given that Parallax offers the source files for the hardware, you could send the files out to have them fabricated in aluminum, carbon fiber, or fiberglass. You can't do that with one of the cheaper frames from Asia.

Most importantly, the Propeller-based HoverFly Open Board provides a powerful open platform for experimentation. That's the real selling point of this quadcopter.

It's designed for the experimenter. Plus, there's no better way to understand a robotics platform than to build it from the ground up. SV

Photo 18. Use prop pitch as a mounting guide. This photo is of the motor at 2 o'clock, intended to spin clockwise.



Connect your robots. Reap the benefits ™

MyRobots.com strives to make cloud robotics a reality accessible to everyone and everything by enabling all robots and smart devices to connect to the Internet.

You can think of MyRobots.com as a social network for robots and smart devices. In the same way humans benefit from socializing, collaborating and sharing, robots can benefit from interacting and sharing their sensors' information, which provides insight on their current state, and allows them to be controlled and monitored remotely.

Through the **MyRobots App Store**, robots augment their capabilities so they can transcend their limitations, allowing them to do more than what they were originally designed for. This means robot owners can reap the full benefits of their robots.

Electronic Messaging With Your Robot

by Fred Eady

Robots don't usually have to visually see something to act on it. For instance, a robot doesn't have to pull out its digital voltmeter to measure a voltage or current. If measuring voltage and current are in the robot's operational domain, the robot is equipped with the proper sensors to sense voltages and currents.

Your "man in a can" is probably not as sophisticated as *Next Generation's* Data. However, if you really break Data down, he's the typical garage robot. He senses things and is programmed to act on them. In Data's case, he can see, hear, smell, and touch. In the end, that's all just simple robotic I/O. Data uses his android senses as data input devices. Your aluminum wannabe android may get its input by way of a serial port, analog-to-digital converter (ADC) input, or SPI portal.

Data produces output by entering values on the ship's console, speaking or performing a physical action. Your mechanical monkey's output can be in the form of controlling a relay, controlling an actuator, or sending a message with its serial port. Then, Data could probably generate and send what we call an email today. Your robot can do that, too.

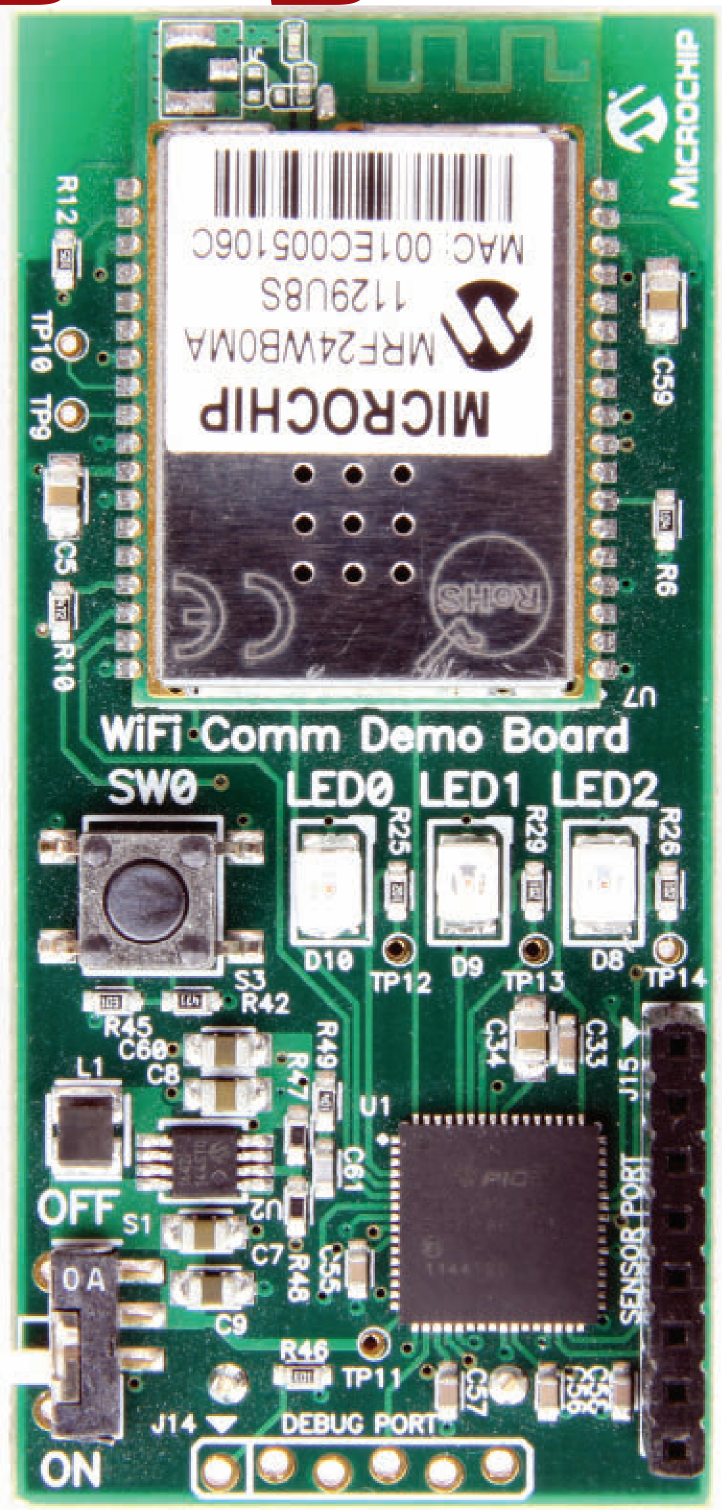
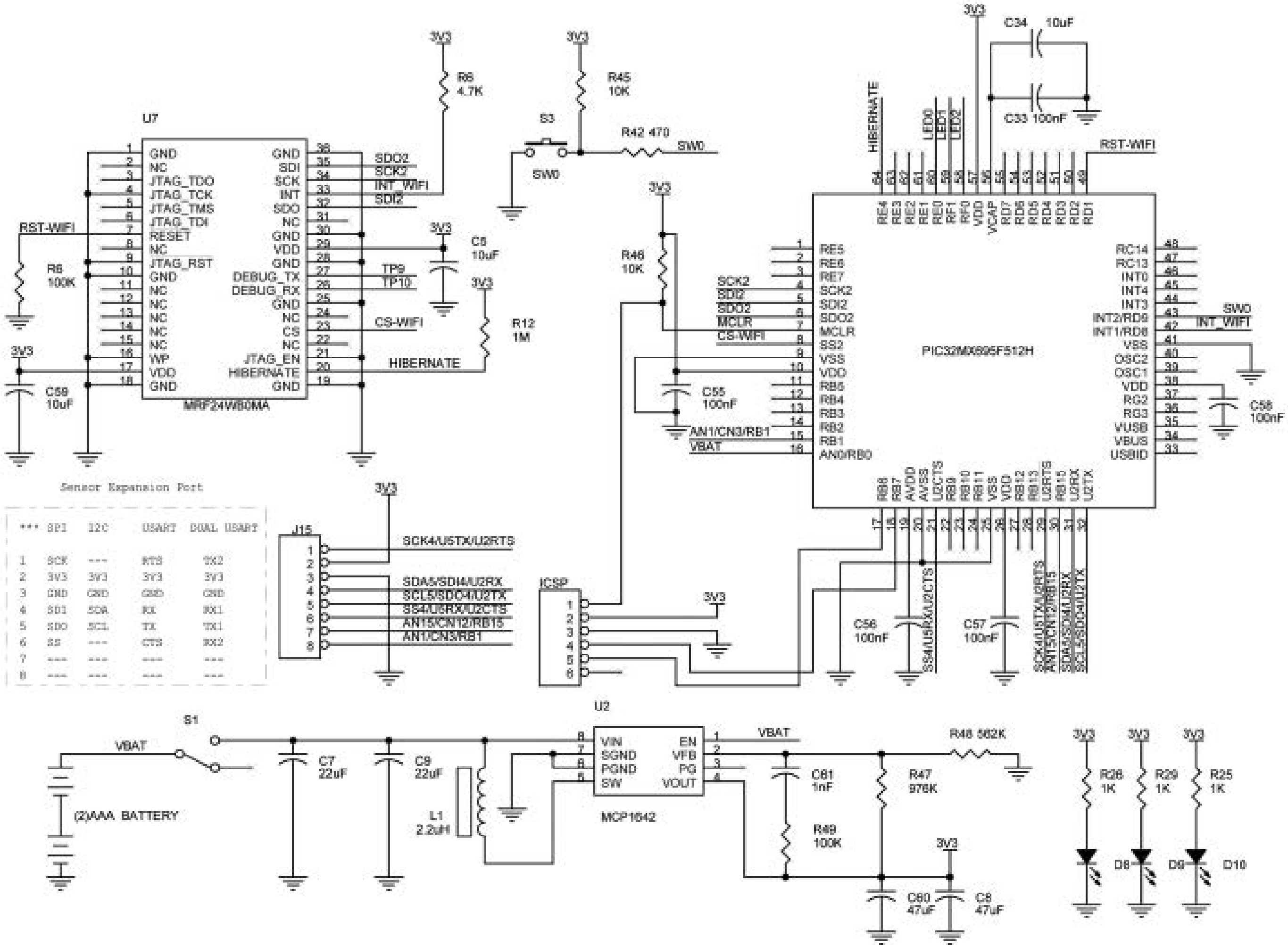


PHOTO 1. The MRF24WB0MA is not as power stingy as the Microchip 802.15.4 radios. However, the MRF24WB0MA's appetite for power can be curbed by forcing it to hibernate.



SCHEMATIC 1. This is a classic implementation of the MRF24WB0MA. Note the absence of any type of crystal.

Robotic SMTP

Roaming robots can keep in touch via the MRF24WB0MA Wi-Fi module.

The Simple Mail Transfer Protocol is designed to be a relatively simple means of transporting mail via the Internet. In the good old days, one could manually send an email using an ASCII terminal and a Telnet session. With the advent of SPAM and all of its wonders and mutations, authenticated email has become the domain of specialized email programs that run on PCs.

While companies like MarshallSoft provide tricky software libraries for C and BASIC, our folks at Microchip are covering email's embedded side. The latest version of the Microchip TCP/IP stack contains code that enables most PIC18, PIC24, and PIC32 class microcontrollers to send open and authenticated email messages.

Microchip goes beyond covering the embedded email model with its microcontrollers. A series of their wired and wireless Ethernet adapters complete the tool set necessary to send an email through the routers and servers that make up the Internet. For stationary robots, the 10Base-T wired email solutions include the ENC28J60 and PIC18F97J60.

Wireless SMTP Hardware

The MRF24WB0MA is easy to implement. However, why would we want to scratch-build our email hardware when a perfectly good ready-to-go platform already exists? The Wi-Fi Comm demo board shown in **Photo 1** is more than capable of assembling and sending an SMTP message.

The Wi-Fi board is based on a PIC32MX695F512H. The only other active components on it are a boost regulator and the MRF24WB0MA. The MCP1642 is a higher current version of the MCP1640 boost regulator that is capable of starting up and working from a single-cell alkaline battery. Microchip plans to release the MCP1642 toward the end of the year. Three LEDs and a pushbutton switch emulate a display and keyboard, respectively. Just enough I/O is exposed on the sensor expansion port to interface an SPI or I^{2}C peripheral and a pair of USARTs. You can get connection details from **Schematic 1**.

Assembling SMTP Firmware

The demo board comes coded to be an HTTP server. The out-of-the-box board forms an adhoc Wi-Fi network and serves a web page sprinkled with little switches and buttons. Obviously, that's not what we want to do. So, the first thing on our wireless email quest is to blow away the Wi-Fi board's stock firmware load.

We'll load up a new TCP/IP stack configuration that includes support for the MRF24WB0MA. However, there is one original piece of firmware that we want to keep and reuse. To save some coding time, we'll recycle the demo board's original *HardwareProfile.h* file. Now, all of the I/O pins associated with the board's LEDs are defined. The *HardwareProfile.h* file also contains the I/O map for its pushbutton switch. As you can see in **Schematic 1**, the MRF24WB0MA is a SPI-driven device. Thus, the SPI hardware pin assignments are included in *HardwareProfile.h*:

```
//——————————————
// MRF24WB0M WiFi I/O pins
//——————————————
#define WF_CS_TRIS            (TRISGbits.TRISG9)
#define WF_CS_IO              (LATGbits.LATG9)
#define WF_SDI_TRIS           (TRISGbits.TRISG7)
#define WF_SCK_TRIS           (TRISGbits.TRISG6)
#define WF_SDO_TRIS           (TRISGbits.TRISG8)
#define WF_RESET_TRIS         (TRISDbits.TRISD1)
#define WF_RESET_IO           (LATDbits.LATD1)
```

This is a good time to show you how fast we're clocking the CPU and peripherals. This set of definitions is located in *HardwareProfile.h*:

```
// PIC32MX processor
#define GetSystemClock()     (40000000ul)    // Hz
#define GetInstructionClock()
       (GetSystemClock()/1)
// Set your divider according to your Peripheral
Bus Frequency configuration fuse setting
#define GetPeripheralClock()
       (GetInstructionClock()/1)
```

With the *HardwareProfile.h* file in place, we can write our initialization function. Keeping with the Microchip conventions, we'll build our initialize code under the *InitializeBoard* function. Since we're dealing with a PIC32MX device, we can code in some system performance and clocking statements unique to PIC32MX microcontrollers:

```
// Enable multi-vectored interrupts
INTEnableSystemMultiVectoredInt();

// Enable optimal performance
    SYSTEMConfigPerformance(GetSystemClock());

// Use 1:1 CPU Core:Peripheral clocks
mOSCSetPBDIV(OSC_PB_DIV_1);
```

Basically, we've enabled the PIC32MX interrupt engine and set up optimal performance using a clock speed of 40 MHz. Earlier in the *HardwareProfile.h* file, we specified the peripheral clock speed equal to the system clock speed. We've made that official with the set peripheral clock function's *OSC_PB_DIV_1* argument.

The more we know about the demo board's microcontroller, the better. So let's take some time to break down the PIC32MX695F512H's configuration words:

```
#pragma config FNOSC = FRCPLL, FPLLIDIV = DIV_2,
FPLLMUL = MUL_20, FPLLODIV = DIV_2, FPBDIV =
DIV_1, FWDTEN = OFF, POSCMOD = OFF, FSOSCEN =
OFF, CP = OFF
```

The Wi-Fi board's PIC32MX695F512H is running on its internal 8 MHz oscillator (FNOSC = FRCPLL). The PLL (Phase Locked Loop) suffix tells us that the PIC32MX695F512H's internal oscillator is being supercharged with the assistance of a PLL. From here, we can do the math to determine how fast the PIC32MX695F512H's CPU is being clocked.

The PIC32MX695F512H internal oscillator runs at a nominal 8 MHz. Its PLL wants to see an input of 4 MHz. Thus, the PLL input is divided by 2 (FPLLIDIV = DIV_2). The next configuration fuse value multiplies the 4 MHz input by 20 (FPLLMUL = MUL_20). The resultant 80 MHz at the output of the PLL is then divided by 2 (FPLLODIV = DIV_2); 40 MHz is applied to the CPU and the PIC32MX695F512H's on-chip peripherals (FPBDIV = DIV_1).

The PIC32MX695F512H is capable of being programmed using its JTAG interface. So, to recapture those I/O pins, we must disable JTAG:

```
DDPCONbits.JTAGEN = 0;
```

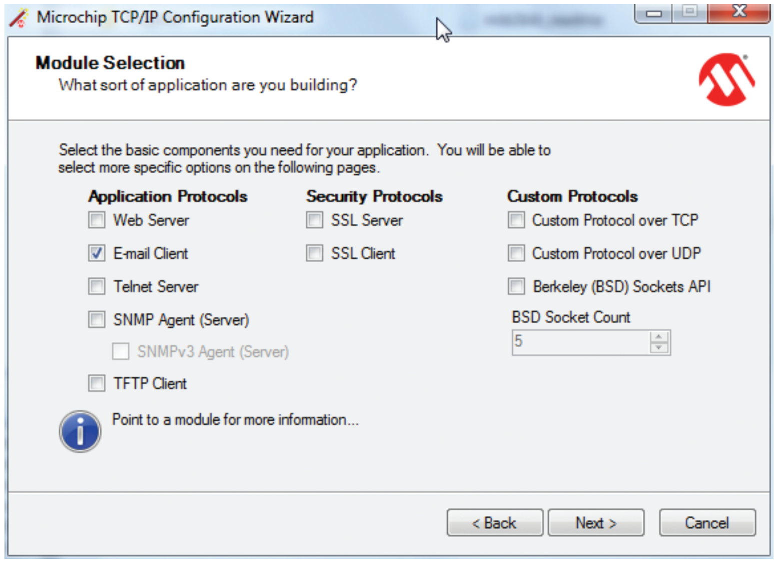
Now we can deal with the LEDs and pushbutton switch:

```
// LEDs
LEDS_OFF();
LED0_TRIS = 0;
LED1_TRIS = 0;
LED2_TRIS = 0;

// Push Button
SW0_TRIS = 1;
```

What's the use in sending an email that just says "Hello?" Let's send some data. How about using a PIC32MX695F512H peripheral library call to turn on its ADC:

```
CloseADC10(); // ensure the ADC is off before
              // setting the configuration

// define setup parameters for OpenADC10
#define PARAM1  ADC_MODULE_ON |
                //Turn module
                ADC_FORMAT_INTG16 |
                //ouput in integer format
                ADC_CLK_MANUAL |
                //trigger mode manual
                ADC_AUTO_SAMPLING_ON
                //enable autosample
```

We control the PIC32MX695F512H's ADC engine by feeding parameters to the OpenADC10 peripheral library



SCREENSHOT 1. The Microchip TCP/IP Configuration Wizard populates the *TCPIPConfig.h* file according to your menu selections.

function. At this point, we've basically told the ADC module to output an integer on our signal.

We're really not interested in the advanced ADC features. However, we do want to make sure that the ADC is referenced to VDD and GND. So let's code that,x plus set the number of samples to be taken and stored in the ADC buffer:

```
#define PARAM2   ADC_VREF_AVDD_AVSS |
                 //ADC ref external
                 ADC_OFFSET_CAL_DISABLE |
                 //disable offset test
                 ADC_SCAN_OFF |
                 //disable scan mode
                 ADC_SAMPLES_PER_INT_2 |
                 //perform 2 samples
                 ADC_ALT_BUF_ON |
                 //use dual buffers
                 ADC_ALT_INPUT_ON
                 //use alternate mode
```

It would be nice to tell the PIC32MX695F512H's ADC where to obtain its conversion clock and how long to sample. While we're at it, we'll squash another one of those advanced ADC features that we don't need:

```
#define PARAM3   ADC_CONV_CLK_INTERNAL_RC |
                 ADC_SAMPLE_TIME_15
#define PARAM4   SKIP_SCAN_ALL
```

Looking back at **Schematic 1**, the ADC input is defined as AN1. With that, we'll enable ADC port AN1, assign it to channel 0, and designate ground as its negative reference:

```
#define PARAM5     ENABLE_AN1_ANA
SetChanADC10(ADC_CH0_NEG_SAMPLEA_NVREF |
ADC_CH0_POS_SAMPLEA_AN1);
```

All that's left to do is pull the trigger on the parameters we specified and enable the ADC engine:

```
OpenADC10( PARAM1, PARAM2, PARAM3, PARAM4,
PARAM5 );
EnableADC10();
```

PARAM1 and PARAM2 are too long for the magazine format and I wanted to show the comments. So, I took the liberty to stack the parameters. In the real world, this will not compile due to the comments separating the OR (|) symbols. In your code, stretch PARAM1 and PARAM2 out in a single row the same way it's done in PARAM3.

Configuring the Microchip TCP/IP Stack

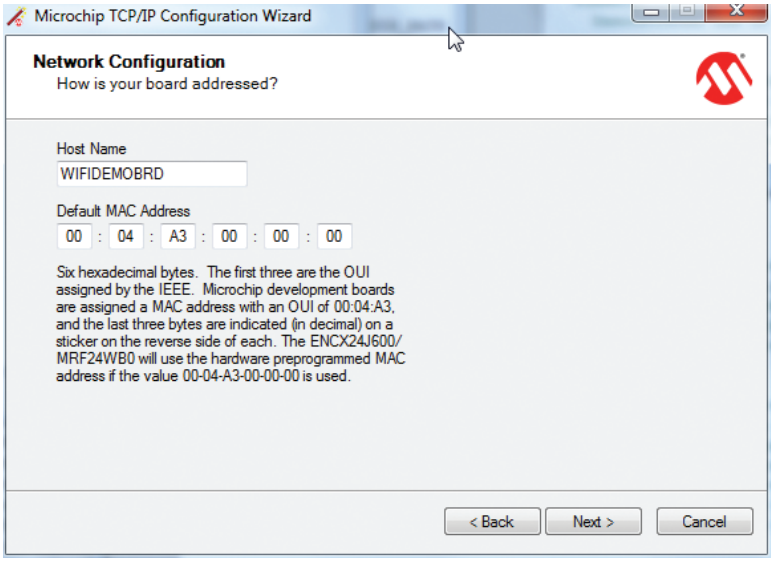
The PIC32MX695F512H side of this project is pretty well in hand. At a minimum, we'll need to do some IP addressing, a userid, and a password. Before we do that, let's configure the TCP/IP stack to do everything we want.

We accomplish this by selecting the necessary stack components in the *TCPIPConfig.h* file:

```
/* Application Level Module Selection
 *   Uncomment or comment the following lines to
 *   enable or disabled the following high-level
 *   application modules.
 */
#define STACK_USE_DHCP_CLIENT
       // Client for obtaining IP address and
       // other parameters
#define STACK_USE_SMTP_CLIENT
       // Simple Mail Transfer Protocol for
       // sending email
#define STACK_USE_DNS
       // Domain Name Service Client for
```

The three components of the TCP/IP stack I've listed are all we need to send emails. The DHCP client helps us get up on the local network and the DNS client helps us find the desired SMTP mail server. The SMTP client authenticates with the targeted SMTP mail server and literally delivers the mail. I like to do the module selection manually because that's how I learned to manipulate the stack. However, you can use the TCP/IP Configuration Wizard which is much easier. Rolling the mouse over the selections in the Wizard activates the context sensitive help messages. **Screenshot 1** shows how to select the SMTP client. Moving further into the TCP/IP Configuration Wizard, we encounter the window shown in **Screenshot 2**. This will populate the Host Name field in the *TCPIPConfig.h* file. The default MAC address is good here as the stack will propagate the MRF24WB0MA's built-in MAC address to the right places in the stack. Here's what the resultant code in the *TCPIPConfig.h* file looks like:

```
#define MY_DEFAULT_HOST_NAME        "WIFIDEMOBRD"

#define MY_DEFAULT_MAC_BYTE1   (0x00)
       // Use the default of 00-04-A3-00-00-00
#define MY_DEFAULT_MAC_BYTE2   (0x04)
       // if using an ENCX24J600, MRF24WB0M, or
```



SCREENSHOT 2. At this point, we're still working on the *TCPIPConfig.h* file. We can keep the default MAC address as the TCP/IP stack will populate the appropriate address fields with the MRF24WB0MA's built-in MAC address.

```
#define MY_DEFAULT_MAC_BYTE3   (0xA3)
       // PIC32MX6XX/7XX internal Ethernet
#define MY_DEFAULT_MAC_BYTE4   (0x00)
       // controller and wish to use the
#define MY_DEFAULT_MAC_BYTE5   (0x00)
       // internal factory programmed MAC
#define MY_DEFAULT_MAC_BYTE6   (0x00)
       // address instead.
```

The Configuration Wizard also supports the wireless component of the TCP/IP stack, whose parameters are found within the *WF_Config.h* file. **Screenshot 3** is the beginning of our wireless network connection. By the time we work our way through the Wizard, the security method will be set and the MRF24WB0MA power control options will be set in place. Clicking on the Wizard's Finish button will file the *WF_Config.h* configuration parameters away. Here's a taste of the real thing:

```
#define MY_DEFAULT_SSID_NAME
       "edtp"
```

```
#define MY_DEFAULT_NETWORK_TYPE
  WF_INFRASTRUCTURE
#define MY_DEFAULT_SCAN_TYPE
  WF_ACTIVE_SCAN
#define MY_DEFAULT_CHANNEL_LIST
  {1,6,11}
#define MY_DEFAULT_WIFI_SECURITY_MODE
  WF_SECURITY_WPA2_WITH_KEY
```

Delivering the Mail

The email delivery process is controlled by the code contained within the TCP/IP stack's *SMTP.c* and *SMTP.h* driver files. The email driver files define and execute a number of state machines. There's a state machine for resolving the email server's address. Once the DNS component of the TCP/IP stack has resolved a valid IP address, another state machine tracks the build and destruction of the socket used to communicate with the email server. A socket is no more than an IP address and port pairing. For instance, an email server socket can be described as a device with an IP address of 265.234.123.005 that services email traffic on its port 25.

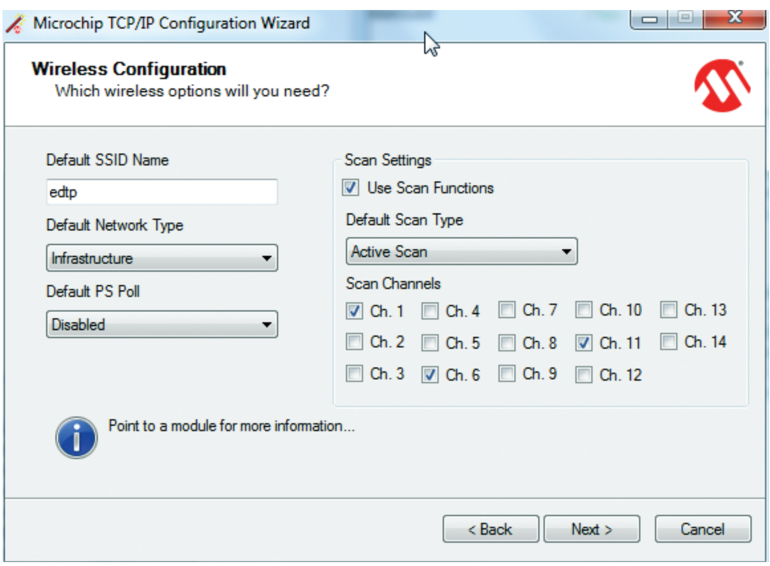
The application layer of the email transmission process is also controlled by a state machine. The state machine pointer variable *MailState* is coded as follows:

```
static enum
{
        MAIL_HOME = 0,
        MAIL_BEGIN,
        MAIL_SMTP_FINISHING,
        MAIL_DONE
} MailState = MAIL_HOME;
```

Your application must kick off the email process in the MAIL_HOME state. Normally, your mechanical creation would logically start the email process. We'll use a physical method in our MAIL_HOME code. We'll push a button, which just happens to be part of the Wi-Fi demo board hardware:

```
switch(MailState)
{
 case MAIL_HOME:
      if(SW0_IO == 0u)
         {
              // Start sending an email
              LED1_IO = 1;
              MailState++;
              LED2_IO = 0;
         }
         break;
```

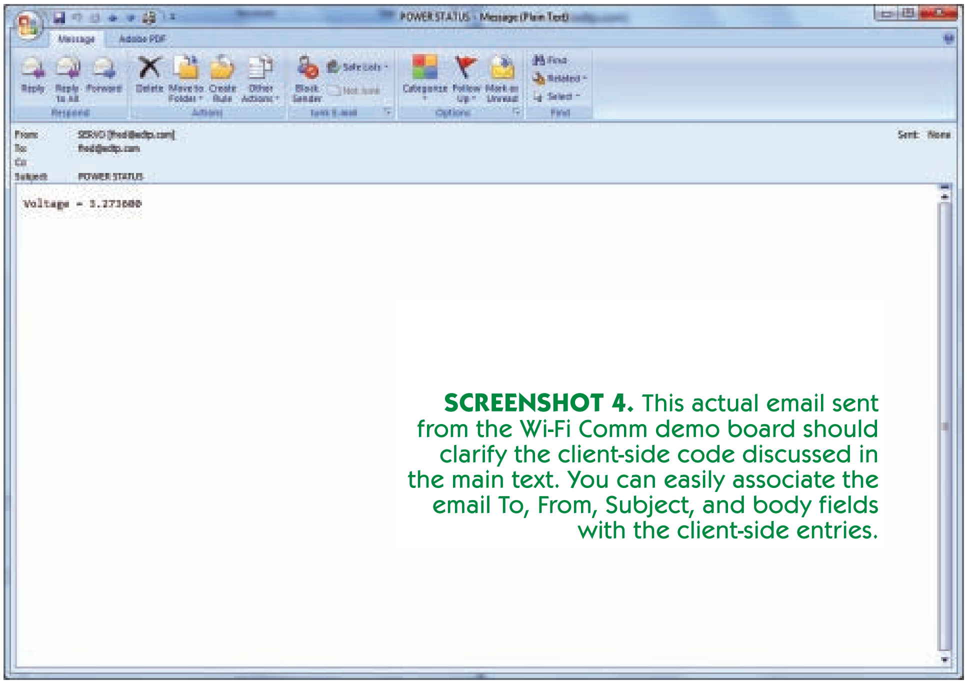
The board's three LEDs provide a visual



SCREENSHOT 3. With this window, we've entered wireless territory. This series of parameters will be placed in the *WF_Config.h* file.

Source

Microchip
Wi-Fi Comm Demo Board
TCP/IP Stack
www.microchip.com



SCREENSHOT 4. This actual email sent from the Wi-Fi Comm demo board should clarify the client-side code discussed in the main text. You can easily associate the email To, From, Subject, and body fields with the client-side entries.

representation of the email state machine's operational status. LED0 blinks continually while LED1 signals the start of the email send process; LED2 denotes the completion of the email transmission.

To use the TCP/IP stack's SMTP services, we must provide the email-related client-side information. You can easily pick out the mail to, email server, user ID, and user password in the client information code that follows:

```
case MAIL_BEGIN:
if(SMTPBeginUsage())
{
  static BYTE RAMStringTo[] = "fred@edtp.com";
  SMTPClient.Server.szROM = (ROM BYTE*)
    "pop.emailserver.com";
  SMTPClient.ROMPointers.Server = 1;
  SMTPClient.Username.szROM = (ROM BYTE*)
      "your_userid";
  SMTPClient.ROMPointers.Username = 1;
  SMTPClient.Password.szROM = (ROM BYTE*)
    "your_password";
  SMTPClient.ROMPointers.Password = 1;
  SMTPClient.To.szRAM = RAMStringTo;
  SMTPClient.From.szROM = (ROM BYTE*)"\"SERVO\"
    <fred@edtp.com>";
  SMTPClient.ROMPointers.From = 1;
  SMTPClient.Subject.szROM = (ROM BYTE*)"POWER
    STATUS";
  SMTPClient.ROMPointers.Subject = 1;
  SMTPClient.Body.szRAM = RAMStringBody;
  SMTPSendMail();
  MailState++;
 }
break;
```

Earlier, we went to lots of trouble to bring up an ADC channel. So, let's deploy some ADC application code which builds the body of the email. Our ADC application code will sample the voltage of an external battery, convert the analog sample to a human-readable voltage value, and place the voltage value into the body of our email message:

```
ConvertADC10();
while(BusyADC10());
battvoltage = ReadADC10(0);
battvoltage *= .0032;
sprintf(ADCString,"%f",battvoltage);

static BYTE RAMStringBody[] = "Voltage =
  xxxxxxxx";
RAMStringBody[sizeof(RAMStringBody)-2] =
  ADCString[7];
RAMStringBody[sizeof(RAMStringBody)-3] =
  ADCString[6];
RAMStringBody[sizeof(RAMStringBody)-4] =
  ADCString[5];
RAMStringBody[sizeof(RAMStringBody)-5] =
  ADCString[4];
RAMStringBody[sizeof(RAMStringBody)-6] =
  ADCString[3];
RAMStringBody[sizeof(RAMStringBody)-7] =
  ADCString[2];
RAMStringBody[sizeof(RAMStringBody)-8] =
  ADCString[1];
RAMStringBody[sizeof(RAMStringBody)-9] =
  ADCString[0];
```

When the message has been successfully transmitted, we'll tear down the house and get ready to build it up again:

```
  case MAIL_SMTP_FINISHING:
       if(!SMTPIsBusy())
       {
              // Finished sending mail
              LED1_IO = 0;
              MailState++;
              WaitTime = TickGet();
              LED2_IO = (SMTPEndUsage() ==
                  SMTP_SUCCESS);
       }
  break;

  case MAIL_DONE:
       if(SW0_IO)
       {
              if(TickGet() - WaitTime >
                TICK_SECOND)
                     MailState = MAIL_HOME;
       }
  break;
}
```

Universal Messenger

Your mechanical animal now has the ability to send email. You don't need any fancy applications to receive the data as any smartphone, laptop, tablet, or desktop PC with email capability can be used as the receiving device. By the way, the email in **Screenshot 4** just came in. **SV**

A Robot Operating System on a Chip

Whether you are new to robotics or are an advanced hobbyist, you probably find some aspects of building a robot difficult, tedious, or time-consuming. The RobotBASIC ROS on a Chip makes the process easier and faster by providing the physical interface for a wide variety of motors and sensors, as well as the required low-level programming.

by John Blankenship
and Samuel Mishal

www.servomagazine.com/index.php?/magazine/article/september2012_Blankenship
Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com

Building a robot can be a daunting task because a wide variety of competencies are required. Physically building a robot requires an understanding of gears and mechanics, along with construction skills involving metal, plastic, and wood. Yet, the physical aspect of building a robot is only the beginning. Once a motorized base has been built, the real work begins.

In order for a robot to interact intelligently with its environment, it must have a variety of sensors. Mobile robots operating in a home or office environment typically need perimeter proximity sensors to detect objects around them, as well as a ranging sensor to measure the distance to objects and walls. It is often advantageous to have an electronic compass to maintain a sense of direction and perhaps even line sensors or beacon detectors.

Most hobby or educational robots are powered by DC motors or servomotors. The monitoring and synchronization of such motors can be greatly improved with optical encoders mounted on the wheels or gear train.

Interfacing such an array of motors and sensors to an embedded microcontroller generally requires a reasonable understanding of electronics. As difficult as hardware interfacing can be though, the software required to make multiple devices work together can be an even greater challenge because the control of motors and the reading of sensors often requires interrupts and/or sensitive timing loops that can easily conflict with other aspects of your program.

The RROS Chip

The RobotBASIC Robot Operating System (RROS) resides in a preprogrammed 25-pin chip. If you use it to build a robot, you do not have to worry about interfacing issues because the RROS handles them for you. The chip can directly drive servomotors and small DC motors (up to one amp). Larger motors (up to 30 amps) are supported using external hardware. Most of the supported sensors connect directly to the chip without additional circuitry.

The list of compatible perimeter sensors includes digital and analog varieties of both infrared and ultrasonic sensors from companies like Sharp, Maxbotix, and Parallax. The system also provides support for a sound transducer, wireless communication, battery monitoring, line or drop-off detectors, a beacon detector, wheel encoders, and even a HMC6352 compass. Having a single chip provide the physical interface for so many devices is extraordinary, but the real power comes from the RROS firmware.

The Firmware

All the time-sensitive code needed to control the motors and obtain and format sensory data is preprogrammed into the RROS chip which must reside in the robot itself. A unique communication protocol allows the decision-making portion of your robot application to be written in the easy-to-use — yet robust — RobotBASIC language running on a PC. RobotBASIC has an integrated

robot simulator that mirrors the sensory configurations supported by the RROS chip.

When the RROS is activated, the high-level commands and functions normally used to control the simulator and read data from its sensors *automatically* communicate over a link with the RROS chip — commanding the real motors and requesting data from the real sensors. When you write a program to control your robot, you can concentrate on *what* needs to be done without having to worry about the details of *how* it actually gets done because the RROS does it for you.

The RROS' ability to manage the robot's resources — while masking the details from the user — makes it a true operating system. For example, the RobotBASIC function rFeel() can be used to obtain the state of various perimeter proximity sensors regardless of what type of sensors are actually used.

This is such an important point, let's look at another example. The raw data obtained from ranging sensors can differ dramatically. Even though PING))) and Maxbotix ranging sensors are both ultrasonic, PING))) sensors provide a reading involving time, while the Maxbotix sensors provide a linear voltage.

Sharp IR ranging sensors are different still, providing a nonlinear voltage that must be transformed to be useful. When the function rRange() is used though, the reading provided will be the distance measured in 1/4 inch units — no matter what sensor is used to make the measurement.

The RROS' ability to manage sensors is equally true for motors. The command rForward 40 moves the simulated robot a distance equal to its diameter. When properly calibrated, the same command will move a real robot a distance equal to its diameter – no matter what type of motors are used or how they are interfaced.

Special rCommands are used to calibrate the RROS so that the real robot mimics — within reason — the operation of the simulator. Simulator-based behaviors that are controlled by sensory feedback correlate surprisingly well with a real robot.

Build Your Robot Your Way

The RROS chip makes it easy to build a robot *your* way. Analyze what you want your robot to do and choose motors that make sense for your application.

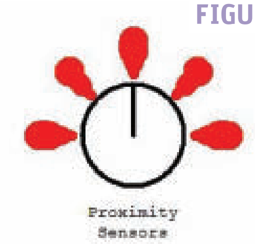
Experiment with the simulator to determine what sensors are appropriate for the situations your robot will be expected to face. Allow your cost limitations, your robot's environment, even your robot's size to dictate your final choices.

Connect your motors and sensors to the RROS chip and it will handle all your interfacing needs, allowing you to program your robot using RobotBASIC's high-level simulator commands. Let's look at a simplified example to illustrate this point.

Creating a Sample Behavior

Assume you want to build a robot that can detect when a wall is encountered using perimeter sensors, and then turn away based on which sensors were triggered. The RobotBASIC simulator has five proximity sensors spaced equally across the front half of the robot as shown in **Figure 1**. Each bit in the sensor reading corresponds to one of the sensors, with the LSB corresponding to the right-most sensor. The robot could respond in many ways, but for this example, let's assume the following behavior:

FIGURE 1.



- The robot should move forward until a wall is detected, then ...
- It should turn left approximately 90º if either of the two right sensors are triggered.
- It should turn right approximately 90º if either of the two left sensors are triggered.
- Approximately means the specified angle plus a random amount up to 30º.

The code in **Figure 2** implements a program on the simulator that creates the desired behavior, and **Figure 3** shows how the simulated robot moves in response to that program. Notice that the code in **Figure 2** is easy to follow because it does not have to deal with the complicated low-level tasks of controlling motors or actually reading sensors.

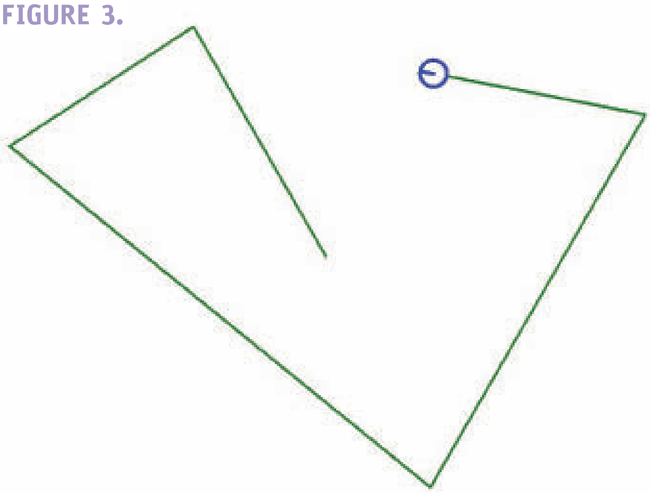
Building the Robot

Now that we've used the simulator to help us develop a program to control a robot, let's see how the RROS can be used to build a robot capable of performing these same

```
gosub Initialization
while 1
  rForward 1
  d = rFeel()
  if d&3    // either of the right sensors
    rTurn -90-Random(30)
  elseif d&24 // either of the left
sensors
    rTurn 90+Random(30)
  endif
wend

Initialization:
  rLocate 400,300,Random(360)
return
```

FIGURE 2.

FIGURE 3.



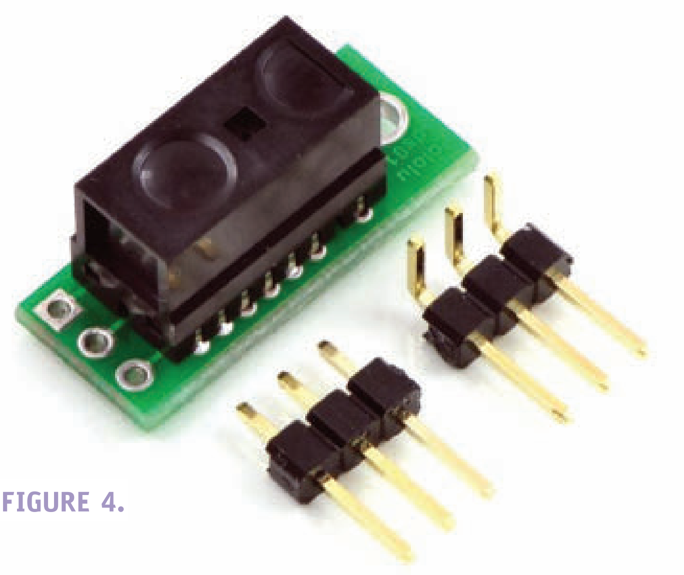


FIGURE 4.

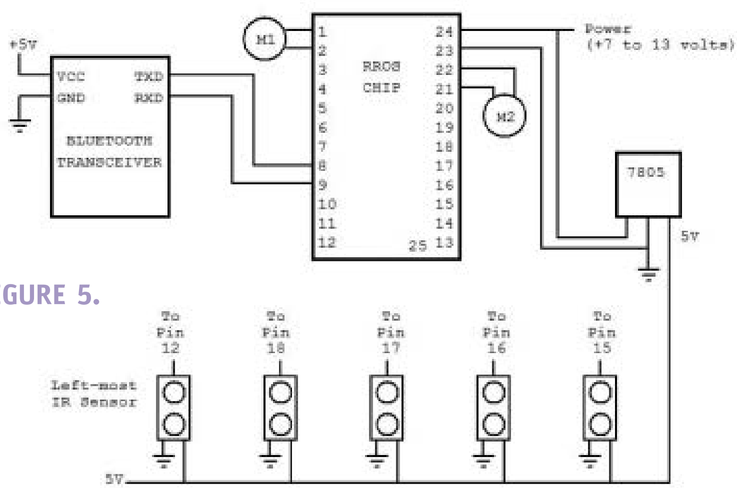


FIGURE 5.

actions. First, connect a Bluetooth transceiver (or other serial communication device), two DC motors, and five digital perimeter sensors (such as the Sharp GP2Y0D810Z0F from Pololu in **Figure 4**) as shown in **Figure 5**.

M1 represents the left motor and M2, the right one. Notice that the only extra circuitry needed is a regulator to provide a five volt supply for most of the components. Mount the motors and perimeter sensors on an appropriate base and you are done. Yes, that is it!

The program in **Figure 2** — with minor modifications — can now control the robot you have just built. There are no programs to compile, no complicated syntax, and nothing to download. You just run the program on a Bluetooth equipped PC and the remote robot will move forward until it detects a wall, then turn away based on the sensors that were triggered — just like the simulation.

The changes that must be made to control the real robot consist primarily of some initialization procedures that inform the RROS chip that DC motors and digital sensors are being used. This can be done by substituting the Initialization subroutine shown in **Figure 6** for the one shown in **Figure 2**.

The first line of the subroutine in **Figure 6** includes a library file that — when called by the next line in the **figure** — defines numerous constants that make it easier to use the RROS. The next line in the subroutine uses the rCommPort command to tell RobotBASIC the number of the serial port used by *your* USB Bluetooth transceiver that communicated with the RROS chip.

Next, the rLocate statement is used to initialize the real robot just as it was previously used to initialize the simulation. Finally, two rCommands are used to tell the RROS chip what motors and sensors are currently in use. Many rCommands are available for calibrating various aspects of the RROS.

With this small change, the program in **Figure 2** will *automatically* communicate with the RROS chip, telling it what actions are expected from the motors and what data is needed from sensors. Once these commands are received, the RROS will control the robot's motors on its own without the need for specific direction from the application program. Sensory data will *automatically* be collected by the RROS and appropriately formatted before it is returned to RobotBASIC. Our example robot used DC motors and digital sensors, but it is important to realize that this same program could be used to control a servomotor-powered robot that used PING))) sensors — or any robot using a supported sensory configuration. Remember too that the RROS does not just manage perimeter sensors. It also oversees tasks dealing with battery monitoring, line sensors, beacon detection, reading a compass, and much, much more.

Limitations

If your robot uses an embedded PC with a wired link to the RROS, your application should be able to read all of the primary sensor values about 40 times per second. If Bluetooth communication is used though, the inherent internal time delays associated with switching between transmit and receive modes reduces the number of sensory reads to about 10 per second. For typical hobby or educational applications, this is generally not a problem because all the time-consuming low-level tasks are being performed in the background for you. Also, special commands have been provided in order to further minimize this potential problem.

For example, when commanded to do so, the RROS can turn the robot to a specified angular orientation based on compass readings or find and face a beacon all on its own. When maximum performance is required though, an embedded PC with a wired link is recommended.

```
Initialization:
  #include "RROScommands.bas"
  gosub RROScommands.bas
  rCommPort 47 //use your port number
  rLocate 0,0
  rCommand(MotorSetup, DC)
  rCommand(SensorSetup, DIGITAL)
return
```

FIGURE 6.

We think novice and seasoned hobbyists can both find advantages to using this approach. If you wish to build a robot this way, you can use our RROS chip (available at **www.RobotBASIC.com**) or study the RobotBASIC interfacing protocol and create your own system. Either way, we suggest downloading a copy of the RROS User's Manual so you can better evaluate if a RROS-based robot can meet your needs. While you are there, download your FREE copy of RobotBASIC. **SV**

Continued from page 21

Wi-Fi Temperature and Humidity Data Logging Sensor

Saelig Company, Inc., introduces the EL-WiFi-TH — a Wi-Fi connected temperature and humidity sensor for monitoring the environment in which it is situated. Data is transmitted wirelessly via a 802.11b-compliant network to a PC, and can be viewed using the free graphical software package supplied. The EL-WiFi-TH can be placed anywhere within range of a chosen network. If the sensor should temporarily lose connectivity with the network, it will log readings until it is able to communicate again with the PC application (max 60 days at 10 second sample intervals). Additionally, the range of the sensor can be increased by using Wi-Fi extenders.

The EL-WiFi-TH is a low-power device that includes a rechargeable internal lithium polymer battery. When configured using typical sampling periods (e.g., once every 60 seconds), the sensor will operate for over one year. The battery can then be recharged via a PC or USB +5V wall adapter using the USB cable provided. The included PC software will allow setup, data logging, and data review for multiple sensors, including immediate graphing of historic data. The EL-WiFi-TH features a built-in LCD screen which can display max and min readings, low battery indication, and Wi-Fi connection strength.

The EL-WiFi-TH is supplied complete with a wall bracket and micro USB cable for $179.95 (software available as a free download).

Saelig Website: **www.saelig.com**

The *SERVO* Webstore

Attention Subscribers ask about your discount on prices marked with an *

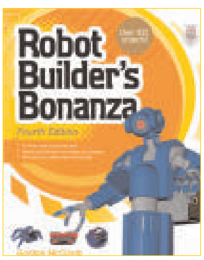
CD-ROM SPECIALS

NEW RELEASE

ROBOTICS

Robot Builder's Bonanza, Fourth Edition

by Gordon McComb



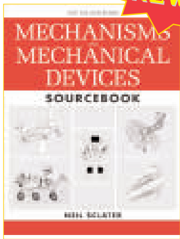
Robot Builder's Bonanza, Fourth Edition includes step-by-step plans for a number of motorized platforms. The book is described as a compendium of robotics topics, containing more than 100 projects, including 10 robot designs new to the fourth edition. These modular robots are low cost, and are made to be reproduced by readers with no training in mechanical construction.

$29.95*

Mechanisms and Mechanical Devices Sourcebook 5th Edition

NEW!

by Neil Sclater



Fully revised throughout, this abundantly illustrated reference describes proven mechanisms and mechanical devices. Each illustration represents a design concept that can easily be recycled for use in new or modified mechanical, electromechanical, or mechatronic products. Tutorials on the basics of mechanisms and motion control systems introduce you to those subjects or act as a refresher.

Reg $89.95 Sale Price $79.95

Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists

by Dustyn Roberts

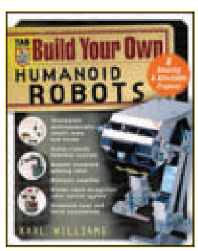
In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.

$29.95*

Build Your Own Humanoid Robots

by Karl Williams

GREAT 'DROIDS, INDEED!

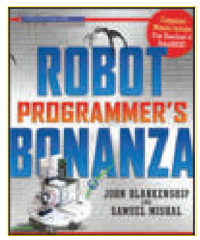


This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot.

$24.95*

Robot Programmer's Bonanza

by John Blankenship, Samuel Mishal



The first hands-on programming guide for today's robot hobbyist!

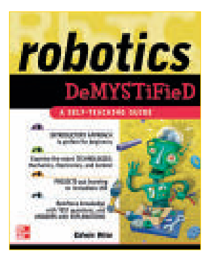
Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.

$29.95

Robotics Demystified

by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!



Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

To order call 1-800-783-4624

SERVO Magazine Bundles

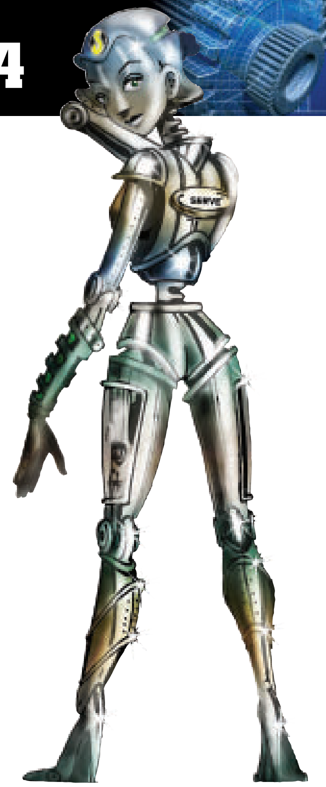
Save $10.00
Only $57.95!

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, 09, 10 and 2011.

Any bot builders out there? Get cool robotics stuff from my store!

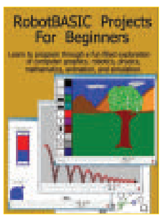
Call me at my order desk!

Visit my online store @ www.servomagazine.com



RobotBASIC Projects For Beginners

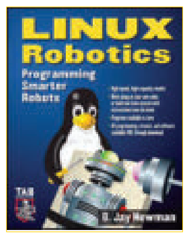
by John Blankenship, Samuel Mishal



If you want to learn how to program, this is the book for you. Most texts on programming offer dry, boring examples that are difficult to follow. In this book, a wide variety of interesting and relevant subjects are explored using a problem-solving methodology that develops logical thinking skills while making learning fun. RobotBASIC is an easy-to-use computer language available for any Windows-based PC and is used throughout the text.
Price $14.95

Linux Robotics

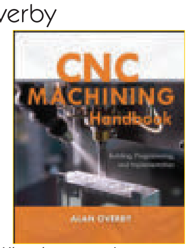
by D. Jay Newman



If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.
$34.95

CNC Machining Handbook: Building, Programming, and Implementation

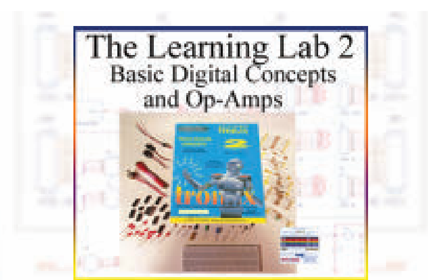
by Alan Overby



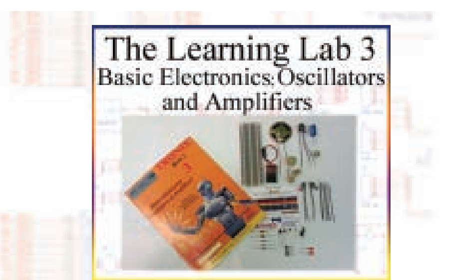
The *CNC Machining Handbook* describes the steps involved in building a CNC machine and successfully implementing it in a real world application. Helpful photos and illustrations are featured throughout. Whether you're a student, hobbyist, or business owner looking to move from a manual manufacturing process to the accuracy and repeatability of what CNC has to offer, you'll benefit from the in-depth information in this comprehensive resource. **$34.95**

FOR BEGINNER BOT BUILDERS

$59.95



$49.95



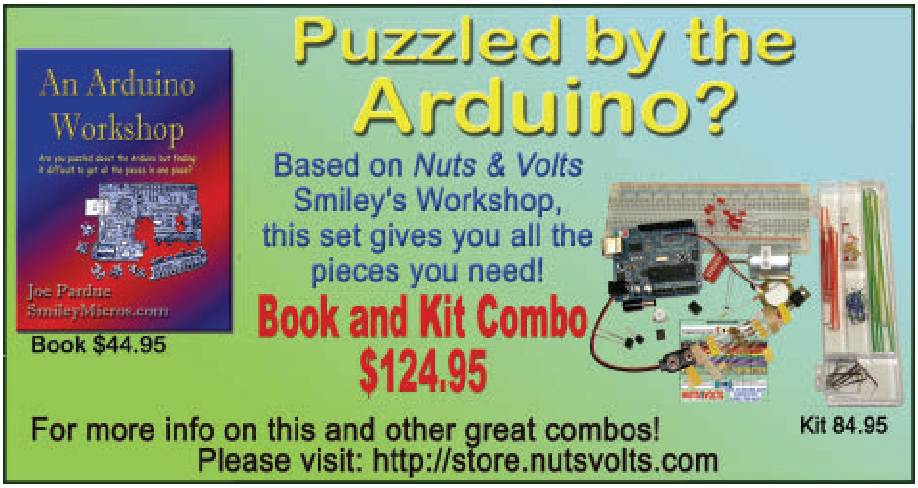
$39.95

The labs in this series — from GSS Tech Ed — show simple and interesting experiments and lessons, all done on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit, and continue to build your arsenal of knowledge with each successive experiment.

For more info and a promotional video, please visit our webstore.

Or order online www.servomagazine.com

SPECIAL OFFERS



Forbidden LEGO

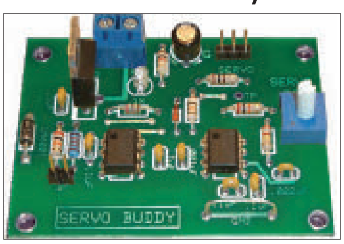
by Ulrik Pilegaard / Mike Dooley

Forbidden LEGO introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse.

Reg $24.95 Sale Price $19.95

PROJECTS

The SERVO Buddy Kit



An inexpensive circuit you can build to control a servo without a microcontroller.

For more information, please check out the May 2008 issue or go to the SERVO webstore.

Includes an article reprint.

Subscriber's Price **$39.55**

Non-Subscriber's Price **$43.95**

3D LED Cube Kit

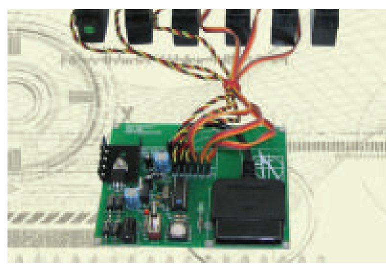
From the article "Build the 3D LED Matrix Cube" as seen in the August 2011 issue of *Nuts & Volts Magazine*.

This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue.
Jig and plastic cases also available.

Subscriber's Price **$57.95**

Non-Subscriber's Price **$59.95**

PS2 Servomotor Controller Kit



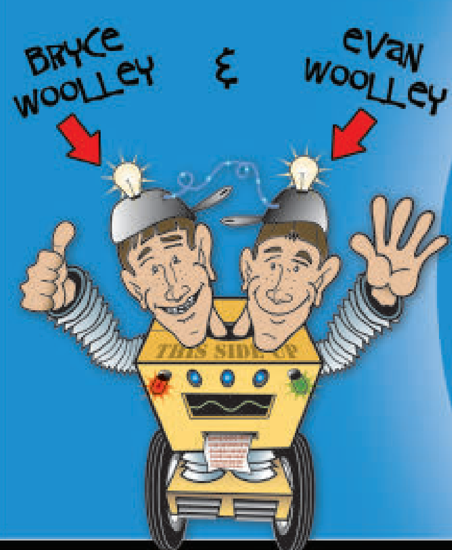
This kit accompanied with your own PlayStation controller will allow you to control up to six servomotors. Includes all components and instruction manual.

For more information, please see the February 2011 edition of *SERVO Magazine*. Assembled units available!

Subscriber's Price **$79.95**

Non-Subscriber's Price **$84.95**

Twin Tweaks

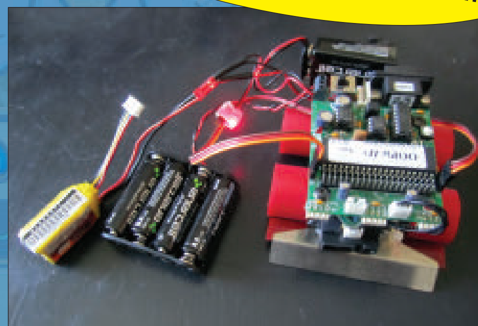


Bryce Woolley & Evan Woolley

THIS MONTH:

The Cobra Strikes Again

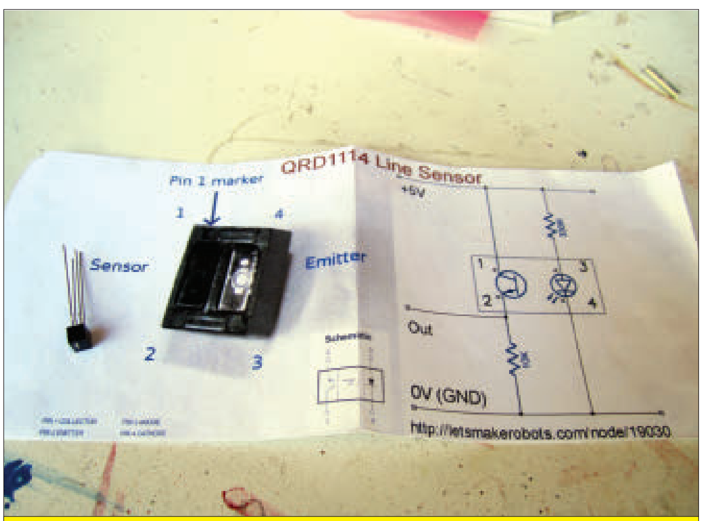
Discuss this article in the SERVO Magazine forums at http://forum.servomagazine.com



THE WIRING SNAKE PIT.

www.servomagazine.com/index.php?/magazine/article/september2012_TwinTweaks

Last time, we had the opportunity to introduce the Cobra chassis from Fingertech Robotics. As with any project, however, our initial endeavors with the Cobra left plenty of room for improvement. The process of optimization is often just as important as the initial design, and we knew that the Cobra chassis deserved more effort on our behalf to unlock the kit's true competitive potential. In particular, we wanted to see if this bot — trained for the mini Sumo dohyo since its inception — would be able to broaden its horizons and serve as an expandable platform for electronics experimentation. Even though Cobras aren't known for their sharp hearing, we thought an ultrasonic sensor would be a great way to see if the Cobra chassis could support more than just a low wedge and a killer drive train.



SNAKE EYES.

The Snake Pit

We think it is important for a roboticist to take pride in their work. This is why we give our robots names like Gog and MO, and it's also a major motivator to take risks, try new designs, and find success inside and outside of competition. With our first crack at the Cobra chassis last time, we were left with a bit of a lumbering behemoth that couldn't escape the feeling of being a first draft.

To review, last time we got our hands on the Cobra mini Sumo chassis. The low profile chassis comes with high friction rubber wheels, a steel and Garolite base, and four Spark gearmotors with a mighty 50:1 ratio. The Cobra chassis is clearly designed to give competitive roboticists an edge in the mini Sumo competition — the chassis is low, heavy, powerful, and has the perfect mini Sumo footprint. We also acquired two TinyESCs — electronic speed controllers for the motors that are necessary when your brain doesn't include onboard motor drivers.

With all of that awesomeness, the Cobra chassis is only missing one thing — a brain! Last time, we transplanted the brain from our Mark III Sumo robot from Junun Robotics. The physical mounting and power requirements were not a perfect match, but we were able to Frankenstein them together and get a final product

about as ungainly as the fictional monster. To Fingertech's credit, the chassis gave our towering, rule-noncompliant Sumo robot solid footing, but we knew that many improvements could be made.

That cliffhanger of sorts is where we left the Cobra last time. The bot was functional, but it certainly wasn't ready to dominate the dohyo. We thought a second look at the bot would be important for a number of reasons. Firstly, there was the aforementioned pride in our work, but secondly we think it is a good reflection of the design process that roboticists engage in whether their project is for a competition or for experimentation. Getting the robot working is never the final step — there are always improvements to be made — and the best competitors and most productive experimenters are often those most skilled at the fine art of optimization.

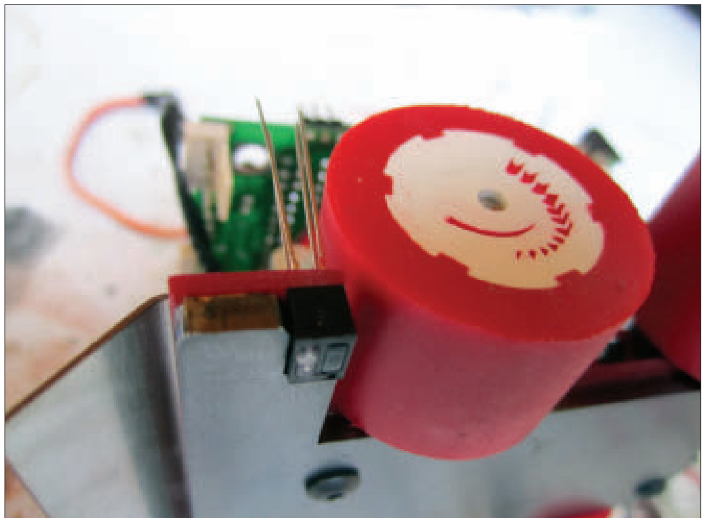
One Battery Pack to Rule Them All

Our first order of business in optimizing our transplant patient was to get the power requirements under control. The current incarnation of our Mark III uses two separate battery packs: a 9V cell for the board electronics and a four pack of AA batteries for the servos. We've all been there. As deadlines loom, we eschew elegant and possibly more effective solutions for something tried and true. This "if it isn't broken then don't fix it" mentality led us to transplant the Mark III's battery packs along with its brain, even though those batteries alone were not sufficient to power the mighty Spark gearmotors on the Cobra chassis.

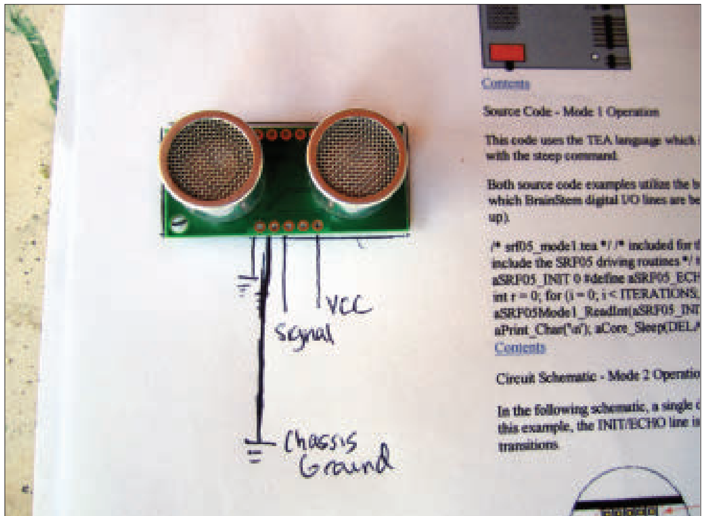
We had to add an extra LiPoly battery pack to give the Spark gearmotors the voltage they hungered for, and that led to an ungainly robot toting three battery packs. This beast of burden certainly wasn't compliant with mini Sumo height or weight requirements, but we justified our decision with that familiar refrain of the initial design: "Well, it works!"

Now that we were taking a second look at the bot, we were determined to cull the herd of battery packs and use only the best pack for the job: the 11.1V lithium polymer "Rhino" battery pack. The Rhino pack is compact and very lightweight — perfect for a weight conscious but power hungry bot. What discouraged us about this battery initially is that it offered far more power than the Mark III actually needed, and we were a bit apprehensive about the dangers of overvoltaging.

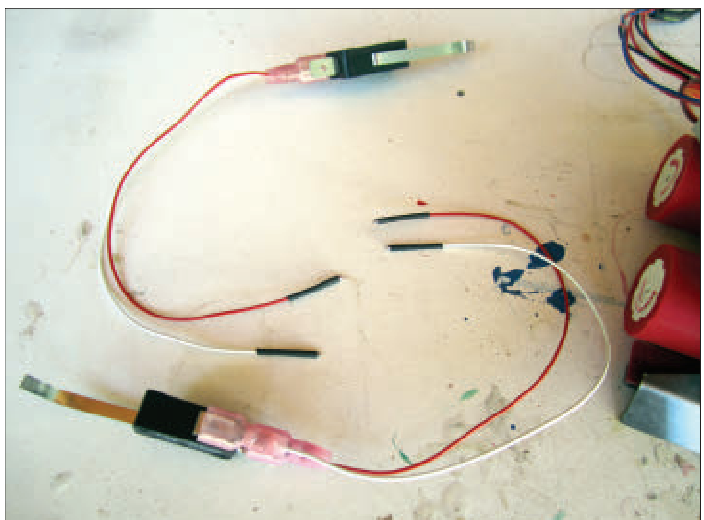
The Mark III used a 9V cell for the board electronics and the four AA battery pack for the drive servos. The AA pack for the drive servos was an easy elimination, though, because now the bot had the four Spark gearmotors which drew power from the LiPoly pack via the Tiny ESCs. After a quick battery-ectomy, we tested the robot and the motors sprang to life, seemingly happy to be free of the unnecessary batteries. This just goes to show that some optimizations can be super simple while still vastly



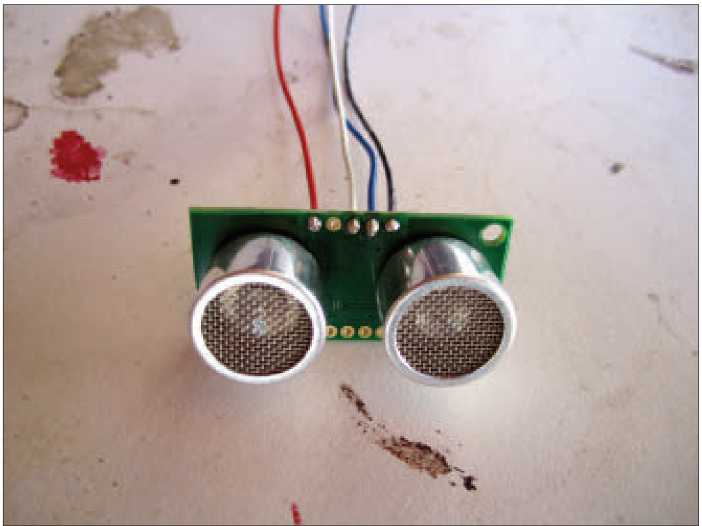
MOUNTING THE LINE FOLLOWING SENSORS.



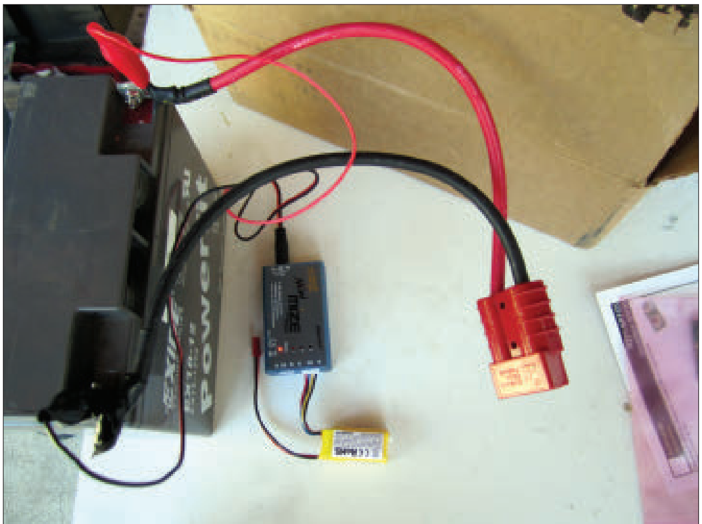
THE SRF-05 ULTRASONIC SENSOR FROM DEVANTECH.



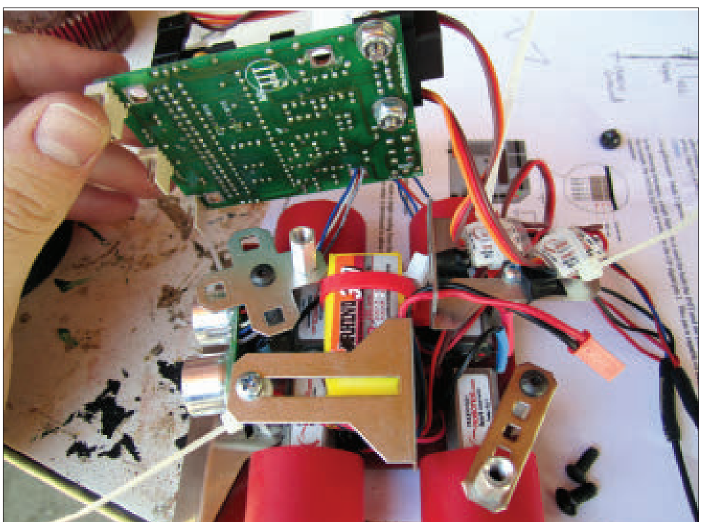
RETRIEVING THE SOCKETS FOR THE ULTRASONIC SENSOR.



Wiring up the SRF-05.



There can only be one ... battery pack.



Prime real estate.

improving performance, and that sometimes the "if it isn't broken then don't fix it mentality" can actually lead you down the primrose path.

With one battery pack eliminated, we still needed to get rid of one more. The 9V battery powered the board electronics for the Mark III brain. We were worried that the higher voltage LiPoly pack might be too much for the board electronics to handle, but a quick look at the board specs revealed the comforting news that the Mark III brain had onboard voltage regulation that could handle up to 16V, making the 11.1 LiPoly pack well within the Goldilocks zone. The only remaining task was to make sure the LiPoly pack could connect to both the board terminal and the TinyESCs.

The LiPoly pack has two connectors branching off from it: a JST connector for the power and ground leads; and a larger header for charging. The header for charging had more leads going into it than we were used to seeing, and it turns out that it is normally charged by a device that both charges and balances. A balancer ensures that all cells are charged to the same state of charge, which ensures that lower capacity cells do not unduly limit the entire pack. This seemed like an appropriately competitive feature for a competitive chassis, so we would do what we needed to ensure that this was the only battery pack the bot needed.

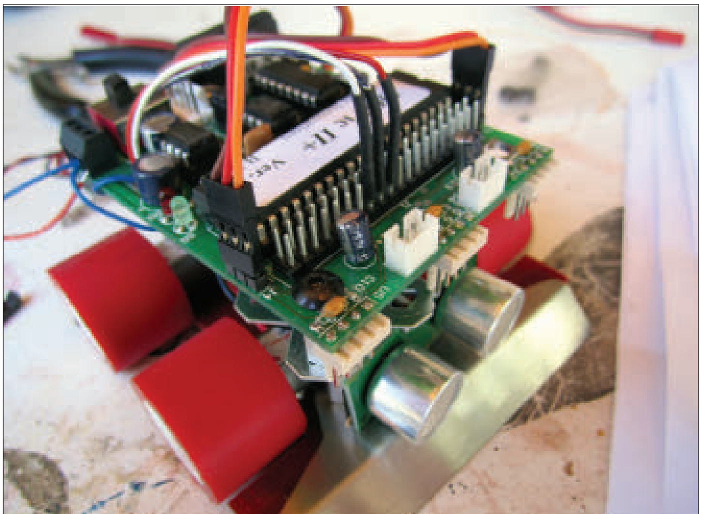
Since the LiPoly pack was capable of meeting all of the bot's power requirements, we just needed to make sure that it was wired up correctly. Last time, we just twisted the power and ground leads from the respective TinyESCs together and stuck them into the JST connector. Now, we also needed leads to go from the JST connector to the terminal block on the Mark III board.

To avoid butchering the wires on the battery pack, we acquired a JST socket for the battery connector to mate to; the JST socket had a power and ground lead to solder the TinyESC leads to and to also add power and ground leads to go to the Mark III board. With this setup, we felt like the wiring on the bot was much more up to our standards of cleanliness, and we were able to store the battery under the brain which was held in place by a bracket fastened to one of the main standoffs protruding from the base.

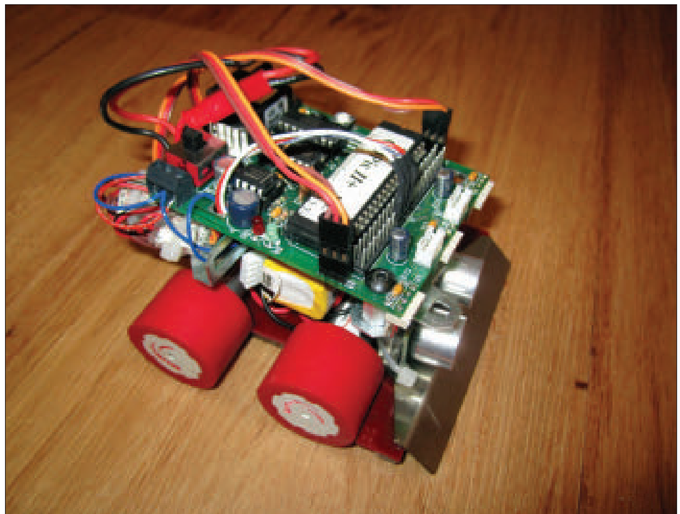
Synesthesia

With the bot down to one battery pack, it was finally starting to look like something that could dominate the dohyo. From what we had seen from the kit so far, its efficacy as a Sumo robot was fairly beyond approach. Serial tinkerers like us, though, are often just as interested in a kit's potential for areas beyond its intended scope. In particular, we were interested in the Cobra's ability to act as a platform for electronics experimentation.

Whenever we come across a new sensor or come up with a new mechanism we would like to test, we usually don't like going through the rigmarole of cobbling



WIRING UP THE SRF-05.



READY FOR THE DOHYO.

together a driving base to carry it around. Having a driving base ready and waiting for such a task allows us to focus on the sensors or mechanism, making the entire process less tedious and more productive.

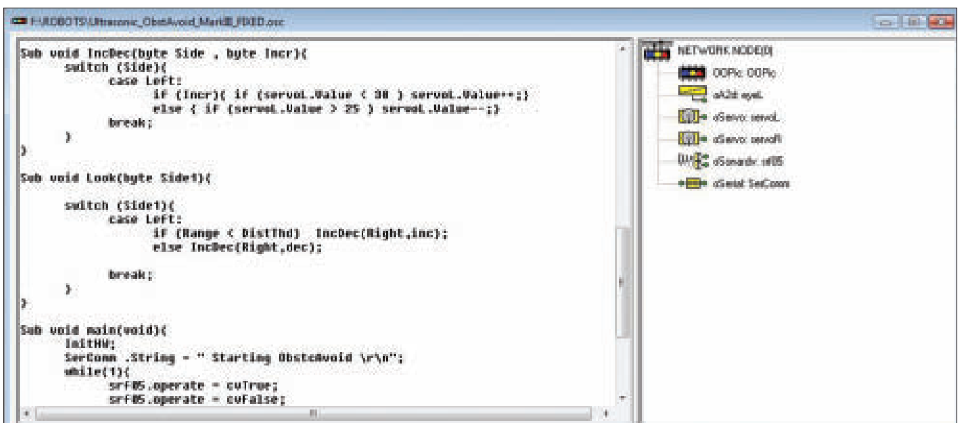
If the Cobra was to be suitable as a platform for experimentation, we could envision some unique advantages based on its design. The Cobra — by its nature — is a competitive bot. The powerful motors and durable base distinguish it from the LEGO or VEX bases that we would often put together when we needed a driveable platform. Especially when a new sensor or mechanism might make its eventual home on a competitive bot, an experimental platform more closely approximating the final product could be beneficial.

With this in mind, we wanted to see if the Cobra chassis could provide such a platform, and we had just the sensor for the job — the SRF-05 ultrasonic sensor from Devantech. We've always thought that ultrasonic sensors have something of an exotic allure to them. IR sensors are a classic solution to giving your bot the ability to avoid (or track) obstacles, and even though the theory behind an ultrasonic sensor is similar on the most general level, giving your bot a sense of hearing carries its own sense of excitement.

The SRF-05 is an upgrade from the popular SRF-04 ultrasonic sensor, and it works by sending out an ultrasonic ping, receiving the reflected ping, and using the time difference to calculate the distance to an obstacle. The SRF-05 improves upon the SRF-04 in a number of ways. It has an increased range of up to 13 feet; it has modest power requirements with a 5V input and only 4 mA operating current; and it can be wired up in a four- or five-wire configuration. In interests of wiring economy, we elected to go with the four-wire configuration which requires a wire for power, one for ground, a second for a chassis ground, and a final wire that we refer to for convenience as the signal wire.

The beauty of the four-wire configuration is that it combines the trigger and the echo into one lead which means one fewer wire winding around your bot, one fewer pin on your controller taken up, and simple programming. The sensor itself has two rows of five solder pads: one row for your soldering pleasure, and one row that was used for initial programming during manufacturing (that should be ignored like Eric Cartman after he's eaten everyone's delicious chicken skins).

We soldered in wires according to the four-wire configuration. For sensors, we often like to put wires into a header to easily connect to the robot's brain. After a quick glance at the schematic for the pins on the Mark III, however, we weren't sure if the pinouts required for the sensor would line up nicely. That was quite all right, because there is a certain type of socket that we love to use on experimenting sensors in particular: individual sockets borrowed from mil spec connectors. Back when



PROGRAMMING THE COBRA.

CAN YOU HEAR ME NOW?

we were first getting started in the world of competitive robotics, we obtained some awesome Maxon motors to use for our combat robots. The motors came with fancy mil spec connectors that featured numerous interlocking pins and sockets. The sockets can be crimped around individual wires to easily connect them to pins on any board. Removing the sockets can be a bit of a chore, so instead we cannibalized the sockets off of some switches that we had used on another snake themed project — the Viper from Microbric (see the September '06 issue for more fun with that modular kit).

With the sensor wired up, we were ready to see if the Cobra chassis could expand its horizons beyond the ascetic minimalism of the mini Sumo ring.

Hear No Evil

Sometimes when robotics kits are meant to be experimental platforms, they will dedicate specific parts of their physical frame to support additions of sensors and mechanisms (see the RobotShop Rover in the March '11 issue for an example). The Cobra has a few such places for sensors. Specifically, there are cutouts in the steel base and holes in the Garolite to accommodate QRD1114 IR sensors. Other than that, however, there are no super obvious places to mount a sensor or additional mechanism.

FEATS OF STRENGTH!

RECOMMENDED WEBSITES

www.fingertechrobotics.com
www.robot-electronics.co.uk/htm/srf05tech.htm

One could conceivably drill additional mounting holes in the base, but that would require keeping the holes in the steel and Garolite aligned. Plus, Swiss-cheesing the baseplate risks jeopardizing the low center of gravity so helpful for a mini Sumo bot. Fortunately, such extreme measures would not need to be taken.

When we first mounted our Mark III brain to the Cobra, we were a bit dismayed to see that the mounting holes on the board did not match up with the standoffs protruding from the base. We were able to rig up an adaptive frame with the VEX pieces that used two diagonally situated standoffs. The others could be used to attach brackets capable of holding sensors. We could also use the standoffs more directly. It made sense for the ultrasonic sensor to face forward, and it turned out that the spacing of the standoffs provided the perfect back support for the sensor. We covered the back of the sensor PCB with electrical tape to avoid shorting anything, and with a few tie wraps we had a set of ears firmly mounted to the front of the bot.

Now that we knew where the sensor felt at home, we could make our final adjustments to the wiring. The mismatch of the standoffs and brain mounting holes ended up being quite serendipitous — the wires for the SRF-05 routed through one of the holes in the brain board perfectly. We also discovered that our initial apprehensions about disparately placed pins were unfounded. All of the pins lined up nicely. We were able to make use of the other standoffs, as well; one supported a bracket that held the TinyESCs and the other held the LiPoly pack in place under the brain. With the bot put together, all we needed to do was program it.

C No Evil

When you're dealing with sensors, wiring them up and mounting them is only half the battle. We still had to program them. Especially when dealing with a new sensor, our tendency is to find example code to build off of. There were plenty of sample codes available on the Internet, but it seemed like almost all of them were in BASIC syntax. We were programming our bot in C.

Looking for another angle, we perused the help file for the OOPIC programming. As it turns out, there was an object dedicated to the SRF-04. The help file also gave example code in BASIC syntax, but once we knew how to call the object we were able to translate the rest. When the program compiled properly, we knew we were on to something, and we set out to test a simple program.

Our simple obstacle avoidance program would instruct the robot to rove around, pinging the ultrasonic sensor. Once the sensor detected an object at a certain range (in our case, we set that range at 30 cm), the robot would turn to the right to avoid the obstacle. We put down some obstacles for the bot to avoid. The Spark gearmotors took the bot careening towards our obstacle, but at 30 cm away the bot took an abrupt turn and went on its jolly way.

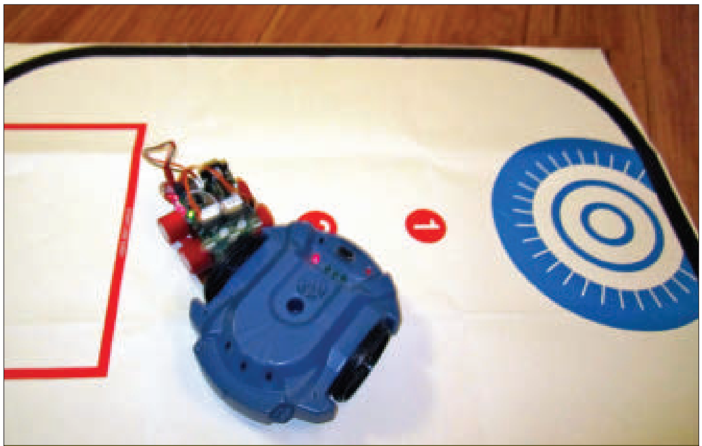
Now that we had the Cobra down to its fighting weight, we also wanted to test its Sumo strength. We started out with some weighty obstacles (books) and an obstacle detection program to ensure that the SRF-05 didn't promote avoidant behavior. The bot pushed around even a stack of heavy books with no problem, and that got us thinking about how this bot had a certain calling that transcended electronics experimentation and pushing around books.

After our frolic with the ultrasonic sensor, we realized that we had not yet used the Cobra chassis for its intended purpose — mini Sumo domination. Unfortunately, its natural foe would be our other mini Sumo robot: the Mark III. The Mark III, however, was missing a few key parts and was in no shape to compete. Fortunately, we had a menagerie of other robots ready to jump into the ring. Calling in particular on our Sensor Olympics competitors, it was no surprise when the Cobra toppled the OLLO bug and scooped the larger Scribbler right out of the field. The combination of the high friction wheels and low center of gravity was deadly indeed.

Cobra Commander

This is all somewhat expected, given the original motivations and goals of the Cobra's design. We got in touch with Fingertech Robotics' Kurtis Wanner to learn more about the genesis of the kit, and to see if our digression with sensor experimentation was an intended use of the kit or simply a happy byproduct.

As it turns out, we think the initial motivation behind the Cobra does help lend the kit nicely to electronics experimentation. The goal of the Cobra was to help take some of the mechanical guesswork out of building a mini Sumo robot for competitors that were more expert in electronic and programming that designing mechanical bits like drive trains.



A RIVALRY FOR THE AGES.

The Cobra was designed to give builders the best mechanical base possible with its powerful motors and low CG. The Cobra was quite a hit with college students in Saskatoon and at a high school level event called myRobotRumble sponsored by the Saskatchewan Institute of Applied Science and Technology (where a Parallax control board paired with the Cobra like a nice red wine and a delicious steak).

We asked Kurtis if the Cobra was also designed with electronics experimentation in mind, and we were mildly surprised that mini Sumo was really the exclusive focus of the kit. The idea of a platform for experimentation conjured up images for Kurtis of large lumbering robots, beasts of burden that could carry around large arrays of data gathering technology. We suppose that such an impression is natural. If we challenge ourselves to visualize an archetypal platform for experimentation, we see something like the behemoths of the DARPA Grand Challenge.

At the same time, though, we think one of the prototypical aspects of an electronics experimentation platform is that it takes all of the guesswork out of the mechanical aspect of the bot, letting the tinkerer focus on electrical and software issues. We feel like that is exactly what the Cobra chassis allowed us to do when figuring out the SRF-05.

Just as the Cobra evolved over the course of these two articles, Kurtis and the folks at Fingertech are intent upon helping the kit continuously improve. In fact, they're already working on pretty much everything that gave us pause as we worked on the kit. They're working on some frame adapter bits to make sure that any brain can be easily accommodated.

Last time, we were also a little wary about the glued-on motors, but we have been assured that the glue sets in a CNC'd jig, and that the setting is super solid. We think that sort of commitment to excellence bodes very well for the success of the Cobra inside and outside of the dohyo. SV

Special Thanks To

Kurtis Wanner

Then and NOW

Sensors for Mobile Robots — Part 4

by Tom Carroll

There comes a time in the minds of every robot experimenter when they just sit back and wonder what would be the ideal robot. I know that I have. Notice that I didn't say what a robot would *look* like or even *act* like. It's what they could *be*, as in a personality of sorts. I never wanted a robot that looked so much like a human that it would fall into the depths of the 'Uncanny Valley,' where it was close to a human likeness but just enough unlike a human to be weird and creepy. Having a robot that looked like Robin Williams in the very first scenes of the movie, *Bicentennial Man,* would be great since Williams had just enough of a robot costume on (as shown in Figure 1) that he was unmistakably a *robot* and not a *human*. I believe we are finally coming a lot closer to this type of robot with smart sensors like Microsoft's Kinect and similar gesture and speech recognition devices. It is these very smart sensors that I will discuss to close this series on sensors.

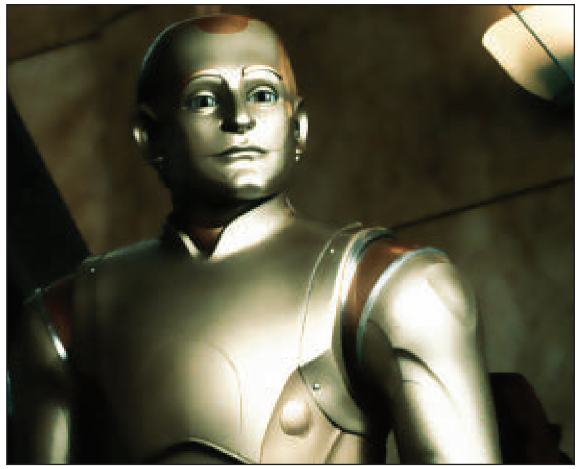


FIGURE 1. Robin Williams as the 'robot' in the film *Bicentennial Man*.

The previous part of my sensor series (in the July '12 issue) covered localization and some specialized types of sensors, such as gas sensors. Robot-specific sensors can range from a simple resistor in series with a motor to measure current and loading, to complex laser imagers, range finders, and similar devices. Today's sensors — especially vision sensors — are many magnitudes better than those of just a few years ago.

I want to cover some of the more unique sensors in this final part, such as visual image and voice recognition sensors. These are not necessarily video cameras that just convert the image of a particular scene into a video signal that a human can view, but are actually intelligent sensors that process an image or verbal command into useful information that a *robot* can utilize for control functions. Facial recognition is another capability of these newer sensors that can give the robot the appearance of having a distinct personality.

Early Speech Recognition for Robots

Let's step back a few years and imagine you live in a 'smart' house. It's 1987 and in this scenario, you unlock the front door of your house and enter. A PIR (passive infrared sensor) is mounted on the far wall and has detected your presence.

Across the room you hear, *"Welcome. Who is this?"*

"It's Joe, Lucy. Please turn on the lights." The black *Mastervoice* box mounted on the wall sends an X-10 signal through the house's wiring to an X-10 receiver module plugged into a wall socket. The table lamp lights up.

"Turn on the TV, Lucy." The TV turns on and warms up (it's a CRT type TV) and you find that a soap opera is on the channel. You hunt for the remote, find it, and flip through some channels to find the news station that you wanted.

"Turn on the sprinkler system, Lucy, and cycle each section through for six minutes each."

You see the first set of sprinklers pop up just outside the living room window.

FIGURE 2. Butler-in-a-Box from Mastervoice.

That black box (shown in **Figure 2**) —now called a Butler-in-a-Box made by Mastervoice — was quite a thrill for techno-nerds back in those days. I personally couldn't afford one of them at $1995, but several thousand were sold for smart homes. X-10 control modules were all the rage back then and are still sold today. The system was envisioned by magician Gus Searcy when a friend noticed that he had to get up to turn on a light and told him that 'a magician shouldn't have to get up to turn on a light.' "As a result," he said, "I decided to create the illusion of an invisible magic genie." The company today is called AVSI and still sells the Butler-in-a-Box, though its quite a bit more expensive now. The box was (and is) too large to be used within a robot, though some robot experimenters have gutted the box and installed the circuitry in their builds.

There were quite a few speech recognition systems available in the mid '80s and most were not particularly functional for robotic needs. The bane of most voice/speech recognition systems has been the need for a microphone close to the speaker's mouth for clear recognition. The Butler-in-a-Box tried to be the exception with the microphone mounted within the black box. Home Automation Living (HAL) was the next try at home automation, with the shameful tie to the computer in *2001: A Space Odyssey*.

We want to be able to communicate verbally with our robots in this same manner. We don't want to be tied to a wired or wireless microphone that we have to wear all the time in order for our machines to hear us correctly. Even now, a head-mounted 'boom' type microphone in front of our lips works best with today's dedicated speech boards and computer-dedicated software packages, such as the Dragon Naturally Speaking voice recognition software shown in **Figure 3**. I have several versions of this package and it works quite well, but I must use the included boom mic for accuracy in speech recognition. Ambient noise is always a problem, and large distances between the pickup microphone and the speaker tend to diminish the higher frequencies that help us distinguish individual spoken words.

I tried (with some success) to use a 'shotgun' style directional microphone on a robot that I built, but it required accurate aiming at the person trying to speak to the robot. Robots don't always face the way we want them to. I used an alarm system PIR sensor to detect a person and one of those solar cigarette lighter parabolic reflectors to focus a person's IR image onto another PIR sensor when

FIGURE 3. Dragon Naturally Speaking speech recognition software.

the microphone was aimed directly at someone. It looked cool but worked miserably.

Speech Recognition for Today's Robots

It's 2012. Imagine the following scenario with your robot in your home. You've parked your car in the driveway and go up a few steps to the front door and place your right thumb over the door lock. It clicks to unlock. You enter the living room. Your personal robot rolls silently up to you; its sensors are detecting your presence via PIR detection and a Kinect sensor mounted in its chest.

"Good evening, Joe. Welcome home. I've deactivated the security system and turned up the A/C to 20 degrees C now that you're home."

"Thanks, Robbie. Key up my favorite Enya album on the MP3 stereo system and then go over there (you're pointing at a corner of the room) *and power down for now."*

"As you wish. Powering down." Robbie rolls over to the corner and powers down to a standby mode. The music from Enya begins to waft through the air.

The Development of Speech Recognition

In this scenario, not only did your robot *visually* and *verbally* recognize you as the owner of the home, he deactivated the security system and turned up the A/C system to the level that he knows that you like. The robot located and started playing the piece of music you desired, and also recognized your hand and finger motions to know where in the room it should go to power down. No handheld remote control was needed and no secret code was typed into a keypad to verify that the correct person

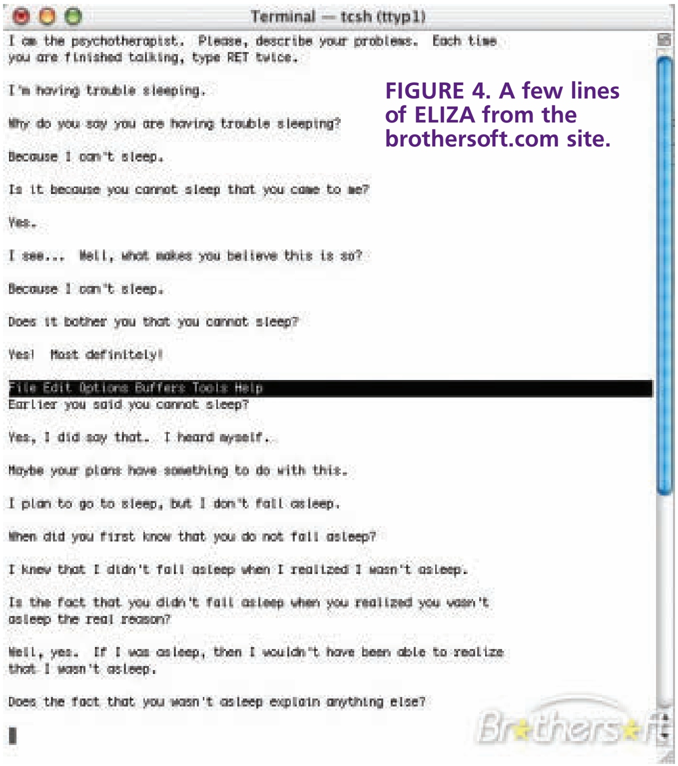


FIGURE 4. A few lines of ELIZA from the brothersoft.com site.

had entered the home and given the commands.

Yes, all of these capabilities are available to today's robot experimenter because of the unique sensors that I'm going to discuss. A robot from a quarter century ago might have had some of these capabilities, but today's lower cost and far more capable sensors make implementation and programming so much easier.

Pseudo-speech Recognition Programs

The old ELIZA computerized psychologist program developed by Joseph Weizenbaum in the mid '60s was a pseudo-speech recognition/AI program that everyone just had to have on their Apple II or early IBM computer. You could type in *"My mother hates me"* and the response would be *"Who else in your family hates you?"* Silly, but intriguing. **Figure 4** shows a few lines of how a doctor/patient conversation might go.

FIGURE 5. The iRobot prototype, AVA.

I saw one computer around 1980 that had ELIZA rigged to a simple voice-to-text peripheral. The results were actually more interesting because the voice recognition device would make many mistakes (think auto-correct) and the resulting psychologist's analysis output was hilarious.

I first saw this program installed on a PDP-11 computer system at Rockwell back in the mid '70s; it took up too much engineering 'play' time, so it disappeared. It was an early example of primitive natural language processing but was far more entertaining than actually being useful.

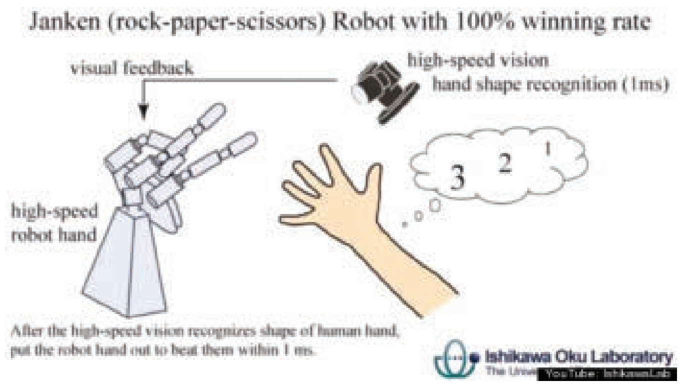
Robots Today With Visual and Verbal Capabilities

The iRobot AVA shown in **Figure 5** made its debut at the CES 2011 show in Las Vegas — the place that all cutting-edge electronic products strive to be seen by the media and industry. The inexpensive modular robot sported an iPad for a head and face, and was demonstrated by iRobot's CEO, Colin Angle. iRobot hasn't been very open about the robot's potential uses, but the early PrimeSense sensor just below the iPad can sense gestures to possibly follow a nurse in a hospital. Angle dodged the questions whether it was a home health care device or just another telepresence robot, but stopped just short of actually revealing the eventual intended application.

Another amazing robot shown in **Figure 6** is what is called the unbeatable Rock-Paper-Scissors opponent. The figure shows paper covering rock for the win. It might be a bit hard to believe, but this robot can beat a human opponent 100% of the time. Developed by the Ishikawa

FIGURE 6. The rock-paper-scissors robot from the IshiKawa Oku Lab at the University of Tokyo.

FIGURE 7. An illustration of the rock-paper-scissors robot's operation.



Oku Lab researchers at the University of Tokyo, a high speed overhead camera (shown in **Figure 7**) controls the mechanical hand that is interconnected through a computer. The camera reads the human's hand movements in one millisecond and quickly responds with the solution that is fed to the mechanical hand. As the lab personnel say, *"In other words, the robot is capable of cheating faster than a human is capable of seeing."*

Smartphone Voice Recognition Apps

With very little new speech recognition hardware being developed during the 2008-2012 time period, smartphone apps took center stage. For example, Siri is an app that seemed to take the technical world by storm in 2011. Shown in **Figure 8** as an ad, it is much more advanced than crude programs like ELIZA. Siri was part of Apple's marketing and is a popular app for all iPhones.

It has been said that "This was more than enough to rejuvenate the speech recognition industry, while putting the pressure back on to create better iterations of speech recognition software."

Google's Voice Search app became available for the iPhone in 2008. This app relies on Google's cloud data center to process voice requests, matching them with the huge pool of human-speech samples and search queries collected by Google. Their newer Android has a similar program called 'the Assistant' that is supposed to be pushed big time in the fourth quarter of 2012.

The Leap Motion Sensor that Recognizes Human Hand Gestures

The little object that looks like a small candy bar in front of the laptop computer in **Figure 9** is actually a motion sensor for computers that is reportedly going to cost only $70. It is called the Leap and is made by Leap Motion — a small San Francisco company founded by Michael Buckwald and David Holz. As they state, "Leap represents an entirely new way to interact with your computers. It's more accurate than a mouse, as reliable as a keyboard, and more sensitive than a touch screen. For the first time, you can control a computer in three dimensions with your natural hand and finger movements. This isn't a game system that roughly maps your hand movements. The Leap technology is 200 times more accurate than anything else on the market — at any price point. Just about the size of a flash drive, the Leap can distinguish your individual fingers and track your movements down to 1/100th of a millimeter."

This unique device caught my eye as the ideal sensor for a small robot since it is a close-up gesture sensor unlike the game-type Kinects by Microsoft with a minimum 40 cm range. The extreme accuracy has been the fodder of many online sites such as **extremetech.com**, and people have assumed that it uses time-of-flight technology that measures the time it takes light to travel to and from an object. The present models shown here and on different sites are just mockups and not actual working prototypes, so the technology is ripe for all types of robot applications. It's due out the end of this year or January 2013.

FIGURE 8. The popular Siri smartphone app.

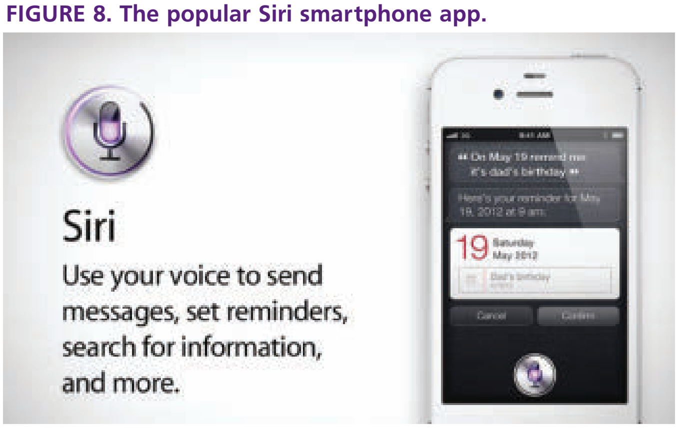


FIGURE 9. The Leap motion control for computers.

FIGURE 10. Early PrimeSense sensors.

The Microsoft Kinect That Started It All

I described some basic features and operation of the original Kinect in this column about a year ago, but the best way to learn anything in detail these days is to use a search engine. There is a tremendous amount of information available. The whole concept of this intelligent sensor was developed by an Israeli company called PrimeSense. Before their systems were developed and improved, most gestural control systems were based on the time-of-flight method I mentioned earlier. PrimeSense's method encodes information in light patterns as it goes out, and the deformation of those patterns is what the camera looks for. Two early examples of their sensor are shown in **Figure 10**. PrimeSense showed off a next-generation TV interface at the 2012 CES that lets you control your TV experience with a wave of your hand. They've also licensed the technology to Asus.

Kinect for Robots

In this year's January and February columns, I wrote about two entirely different robots that both use the Microsoft Kinect sensor. One was the TurtleBot developed at Willow Garage, headed by President and CEO Steve Cousins. Willow Garage develops cutting-edge hardware such as the PR-2 and TurtleBot, and open source software such as ROS for personal robotics applications.

The second robot was Eddie, co-developed by Parallax for the platform, and Microsoft for the RDS (Robotics Developer Studio) software. Both robots are shown in **Figure 11**. Eddie (on the left) is equipped with the newer Kinect for Windows and the TurtleBot (on the right) uses the original Kinect for the Xbox 360 gaming system. Both robots can be made to operate with either of the sensors with some software changes and can also be made to use either the ROS or RDS operating systems.

Kinect for Xbox vs. Kinect for Windows

In the past year, I have used each of these sensors on both robots, as well as bench tested them with TV interconnection. I'll be the first to admit that software and programming are not on the top of the list of my talents. Jessica Ulemen — Engineering Manager at Parallax — was most accommodating to me during my learning curve for the RDS and the SDK (Software Development Kit) as applied to Eddie and the Kinect sensor. The Microsoft team was undergoing some personnel leadership changes within the robotics group, but I was always able to locate and communicate with someone who could steer me in the right direction.

Kinect for Windows (K4W) was not developed for robotic purposes and, in fact, Microsoft never intended the original Kinect to be anything but a game peripheral. As we all know, many people into robotics jumped on this unique sensor as the eyes for their potential robot. The Kinect for Windows is not a typical consumer product like the original, but is a programmer's development tool and is marketed as such. The new Kinect

FIGURE 11. The Parallax Eddie and Willow Garage TurtleBot.

used in conjunction with Kinect for Windows SDK can be used to build applications with C++, C#, or Visual Studio Basic by using Microsoft Visual Studio 2010. The sensor unit does not ship with any software and will only operate with an application developed for Kinect for Windows; it is not for gaming use. Microsoft is looking at a wide variety of business and industrial applications for the new Kinect, but robotics has remained in the forefront of uses.

What's New About the Kinect for Windows?

The differences between the Kinect for Xbox and the new Kinect for Windows is that the K4W includes "a near mode, skeletal tracking control, API (Application Programming Interface) improvements, and improved USB support across a range of Windows computers and Windows-specific 10' acoustic models. The sensor looks identical to the original Kinect but has a shortened USB cable to ensure reliability across a broad range of computers, and includes a small dongle to improve coexistence with other USB peripherals. The newer hardware has different firmware, for one thing. The newer firmware allows depth detection as near as 40 cm. The older firmware only allowed depth detection from 80 cm."

What they're saying here is the original Kinect was designed as a game controller to be used in front of a TV by the person playing a game and was not required to be any closer than 31" away from the sensor.

After trying to figure out the Kinect SDK for Windows 1.5, MS Visual Studio 2010, the APIs, and a few calls to Parallax, I got the K4W working on Eddie. Parallax has been very supportive of Eddie and will take the time to assist a potential customer or purchaser with their questions.

The near mode allows the sensor to be placed at a lower location on the robot, though I ended up replacing it back up to the original position for gesture control from a greater distance. The TurtleBot design team placed the original Kinect 11" above the floor level and at the back of the robot to allow floor detection (80+ cm away) to above head height detection. I can see industrial applications with the K4W near mode as it seems to be somewhat similar in operation to the Leap sensor we talked about earlier.

Some people have remarked that they wished the Kinect could be powered through the USB plug like the similar Asus unit shown in **Figure 12**. I had a very difficult time locating knowledgeable personnel at Asus in the US office or in Taiwan who knew the technical aspects of their product, though it appears to be as capable as either of the two Microsoft units. As such, I was not able to actually test and evaluate one of the Asus units.

The giant from Redmond, WA is the company we all love to bash. With sales of 67 million Xbox 360s and 19 million Kinect sensors, Microsoft will continue to support this unique sensor that many of us have added to our robots. With over $56 billion in sales attributed to these two products, they are not about to drop their support. The competent technical people that I contacted both at Microsoft and Parallax were more than helpful in assisting me. Despite the need for an external 12 volt power source for both of the Kinect sensors and the extra cost for the K4W to cover the non-commercial licensing aspects, these sensors will find their way into many intelligent vision applications beyond mobile robots.

FIGURE 12. Asus Xtion sensor.

Final Thoughts

Microsoft Research is actively working on other technology such as the use of a laptop's microphone and speakers to develop a Kinect-like sonar system called SoundWave which will detect hand motions without additional hardware. They have also demonstrated a prototype of a Kinect-enabled shopping cart for Whole Foods in conjunction with a Texas company, Chaotic Moon. The cart will follow a customer through a store and scan each product placed in the cart.

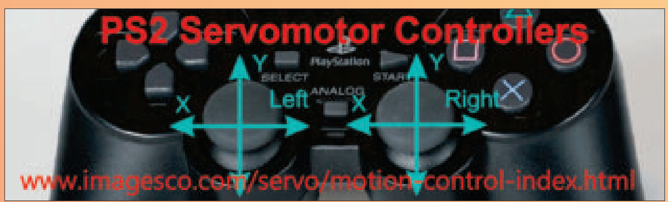
OpenKinect is a group dedicated to making use of the Kinect hardware to enable the device to be used with Windows, Linux, and Macs. The Computer Science and Artificial Laboratory at MIT has added a $150 Kinect to a $400,000 Willow Garage PR-2 as shown in **Figure 13**.

Kinect, Siri, and other vision and speech recognition products are expanding the capabilities of robots and others applications around the world. **SV**

Tom Carroll can be reached at TWCarroll@aol.com.

FIGURE 13. PR-2 with Kinect (courtesy of CNET.com).

ROBO-LINKS



Got electronics?
www.nutsvolts.com

For the finest in robots, parts, and services, go to www.servomagazine.com and click on Robo-Links.

Advertiser Index

Life with Robot!
Not a dream anymore

THE MASTERPIECE
of ROBOT SERVO

24kgf.cm @ 7.4V
[333.6 ozf.in.]

Developed with the ease of use in mind, Herkulex Smart Servo Series are the best 'Green' energy efficient smart servos in the world with low voltage, high output, asynchronous bi-directional communication, and more than 50 set-up parameters.

HerkuleX

Specification (DRS-0101, DRS-0201)

- **Dimensions** : 44.5mm(W) X 24.0mm(D) X 32.0mm(H) [1.75in. X 0.94in. X 1.26in.]
- **Weight** : 45g (DRS-0101) / 60g (DRS-0201) [1.59oz (DRS-0101) / 2.12oz (DRS-0201)]
- **Input Voltage** : 7.4V DC (DRS-0101) / 7~12V DC (Optimized 7.4V) (DRS-0201)
- **Stall Torque** : 12kgf.cm@7.4V (DRS-0101) / 24kgf.cm@7.4V (DRS-0201)
  [166.8 ozf.in. (DRS-0101) / 333.6 ozf.in. (DRS-0201)]
- **Maximum Speed** : 0.166s/60 ° @7.4V (DRS-0101) / 0.147s/60 ° @7.4V (DRS-0201)
- **Operating Angle** : 320 °, Continuous Rotation
- **Communication** : Full Duplex Asynchronous Serial(TTL), Multi Drop, 0-254 ID, Maximum Baud Rate : 0.67Mbps
- **Motor** : Metal Brush DC Cored (DRS-0101) / Coreless DC (DRS-0201)
- **Gear** : Super Engineering Plastic (DRS-0101) / Reinforced Metal (DRS-0201)
- **Feedback** : Position, Speed, Temperature, Load, Voltage, etc.
- **Features** : PID, Feedforward, Trapezoidal Velocity Profile, Velocity Override, Torque Saturator & Offset, Overload Protection, Neutral Calibration, Deadband, 54 Selectable Setting Parameters (Sold Separately : HerkuleX Manager Kit)

HerkuleX Manager is a bundled software that uses the GUI to maximize the ease of operation in setting up more than 50 servo operating parameters and servo maintenance using such a tool as the real time trend graph.

HOVIS Lite　HOVIS Genie　HOVIS Eco

Dongbu Robot

Dongbu Robot Co., Ltd. www.dongburobot.com

Headquarter : 11th Floor, Bucheon Techno Park 401, Yakdae-dong, Wonmi-gu, Bucheon-city, Gyunggi-do 420-734, Korea
TEL : +82-32-329-5551(ext.311), FAX : +82-32-329-5569, E-MAIL : robotsales@dongburobot.com
Factory : 27, 6 Gil, 4 Sandan, Jiksan-eup, Seobuk-gu, Cheonan-city, Chungcheongnam-do 331-814, Korea, TEL : +82-41-590-1700, FAX : +82-41-590-1701


SERVOCITY
New Product!
NEW!
Quarter scale
HITEC
SERVOblocks™
BY ROBOTZONE
Patented
Compatible.
The new Quarter Scale ServoBlocks™ are compatible with the following Hitec servos: HS-755MG, HS-755HB, HS-765HB, HS-785HB.
Functional.
When used with a Hitec HS-785HB servo, the user can achieve 3.5 rotations without any modification.
Versatile.
The extreme versatility of ServoBlocks™ allow users to create complex, extremely rigid, structures with ease using quarter-scale Hitec servos.
Features:
- 1/2" aluminum hub shaft
- 6061 T6 aluminum framework
- Endless attachment possibilities
- Easy to assemble
- Increase lateral servo strength!
Find us on:
www.youtube.com/ServoCity
www.facebook.com/ServoCity
Twitter: @ServoCity
ServoBlocks™ are a must have for any project that demands the most out of your servo!
One online store. Thousands of products. Millions of possibilities!
Have a smartphone?
Scan the QR code to watch a video of ServoBlocks™ in action!
Servos
Radio Systems
Pan & Tilts
Electronics
Wiring
Gears & Sprockets
(620) 221-0123
sales@servocity.com
To view our entire selection of products please visit us online at
www.SERVOCITY.com
Specializing in servos, radios, motors, pan & tilt systems, batteries, wiring, robot kits, hardware, electronics, actuators, gearboxes, and much more!